RADIK:
AN INTERACTIVE GRAPHICS AND TEXT EDITOR

Richard F. Ashford

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

RADIK:
AN INTERACTIVE GRAPHICS AND TEXT EDITOR

by

Richard F. Ashford, Jr.

Thesis Advisor: G. A. Rahe

June 1972

Approved for public release; distribution unlimited.

RADIK: An Interactive Graphics and Text Editor

by

Richard F. Ashford, Jr.
Lieutenant, United States Navy
B.S., Auburn University, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1972

ABSTRACT

RADIK is an interactive graphics and text editing system designed for use with an ADAGE AGT/10 graphics computer, either in a stand-alone mode, or in conjunction with an XDS 9300 computer. The thesis presents an overview of desirable attributes and capabilities of an interactive graphics display system. A description is given of the graphics display system presently in use at the Naval Postgraduate School Computer Laboratory, along with its apparent deficiencies. Objectives for an improved graphics and text editor are presented, in addition to results achieved and problems encountered while designing RADIK. A brief summary of results and applications is presented and implementation of RADIK is proposed. Computer programs developed during the work are appended for reference.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

The author wishes to express his appreciation to Mr. William Thomas of the Electrical Engineering Computer Laboratory for giving freely of his time and assistance throughout the development of the work presented in this thesis.

I. INTRODUCTION

The aim of any computer graphics system is to maximize the information transfer between the user and the computer, preferably in an interactive way. As the applications for computer graphics displays expand, characteristics of the "optimum" system become more difficult to establish. "Optimum" characteristics are a function of the problem to be solved, i.e., the application. But the user may be further constrained by budget limitations or equipment availability. At the same time, the equipment manufacturer, although he would like to offer the "optimum" system to each user, is more often forced to produce standardized, versatile configurations which will satisfy many requirements. Frequently then, the user may find his graphics system lacking in one or more attributes which would be desirable for his application.

Attributes which are typical of most computer graphics systems are [1]:

(1) Generation of a steady, non-flickering display.

(2) Generation of typical graphic elements (e.g., characters, lines, cursors, etc.).

(3) Performance of certain "routine" editing functions - such as light pen tracking, erasing, scaling, rotation, and translation.

The functions listed in attribute (3) can be accomplished with hardware, software, or a combination of both hardware and

software. Those systems which utilize software for performing the graphics editing function afford the user an opportunity to "redesign" or modify the existing editor so that it more closely resembles his "optimum" system. Such a system exists in the Computer Laboratory at the Naval Postgraduate School.

The purpose of this thesis is threefold:

(1) To discuss design considerations taken into account in redesigning GATED, the graphics and text editor presently in use at the Naval Postgraduate School;

(2) To present RADIK, a graphics and text editor which more fully utilizes the capabilities of existing hardware at the Naval Postgraduate School; and

(3) To propose implementation of RADIK at the Naval Postgraduate School.

The thesis is divided into five parts. First, a brief background of graphics research is presented. Secondly, this thesis deals with the considerations taken into account when designing RADIK. This includes a description of the hardware installed in the computer laboratory of the Naval Postgraduate School, along with the present editing system, GATED, and its apparent shortcomings and deficiencies. In addition, the second section contains a discussion of the XDS 9300 FORTRAN graphics package, a set of FORTRAN callable subroutines which are used to create graphics data blocks and to transmit them to the display computer.

The third part of the thesis presents RADIK, a software graphics and text editor which can be used at the Naval Postgraduate School. RADIK is compared to GATED and objectives are stated. Results of the work on which this thesis is based are then presented.

The fourth section summarizes the work and the fifth proposes implementation of RADIK at the Naval Postgraduate School. Program listings which were developed during the design of RADIK are appended for further reference.

II. BACKGROUND

Once computer graphics systems had become a physical reality, most of the technological research in the field was devoted to providing improved hardware devices--devices which could operate more reliably and with higher quality output--for printing and plotting information. Very little was done to produce sophisticated hardware and software techniques to better match the user to the computer. In the past few years, however, the emphasis on visual displays has channeled much research toward improving interaction between man and the machine. In the early stages of this research, a great deal of attention was given to the problem of standardizing programming languages and developing economical, efficient compilers to translate from programming languages to machine code. Such work still continues. However, more emphasis is now being placed on the development of graphics languages.

Much work has been done and many papers been written on the design of a general purpose graphics language ([2], [3], [4], [5]). However, the designer of a general purpose graphics language is faced with a variety of users, each with his own special applications. In order to satisfy the greatest number of needs, the designer is forced to create a language which will not satisfy some needs, and is, therefore, not generalized. One suggestion is to design a "super-language", which would contain subsets for implementation by the various users.

The design of such a language, of course, would have to take into account the fact that different users also have different hardware. We are faced with the alternatives of a special purpose graphics language for each user or, perhaps, several "semi-general" purpose graphics languages, each of which would accommodate many users. While the work presented in this thesis was not in the area of language construction or language design, it should be noted that RADIK provides the interactive capabilities which most languages strive for, while requiring the user to use only FORTRAN programming and function switches to implement them.

III. CONSIDERATIONS IN DESIGNING A GRAPHICS AND TEXT EDITOR

A. EXISTING GRAPHICS HARDWARE AT THE NAVAL POSTGRADUATE SCHOOL

In order to understand the problems encountered in attempting to design a graphics and text editor, one must be aware of the philosophy, data structures, operating procedures, and hardware configuration of the system in which it is to reside. Reference [6], describes the Computer Laboratory in detail. Present equipment at the Naval Postgraduate School includes four computer systems: a medium size, general purpose, digital computer system (XDS 9300), two small graphics computer systems (AGT/10), and one analog computer system (CI 5000). These systems may be used separately (referred to as stand alone operation) or in combination with one or more of the other systems. Figure 1 shows each system and its communication links to the other computer systems. Note that the analog and graphics systems each communicate with the XDS 9300, but cannot communicate with each other except via the XDS 9300.

1. XDS 9300

The XDS 9300 is the main computer in the laboratory. It has a main (core) memory of 32K words and a magnetic drum for secondary storage. System operation is controlled at the operator's console and teletypewriter. Input to the XDS 9300 is via card reader, teletypewriter, paper tape reader, or magnetic tape drive while output may be sent to the line

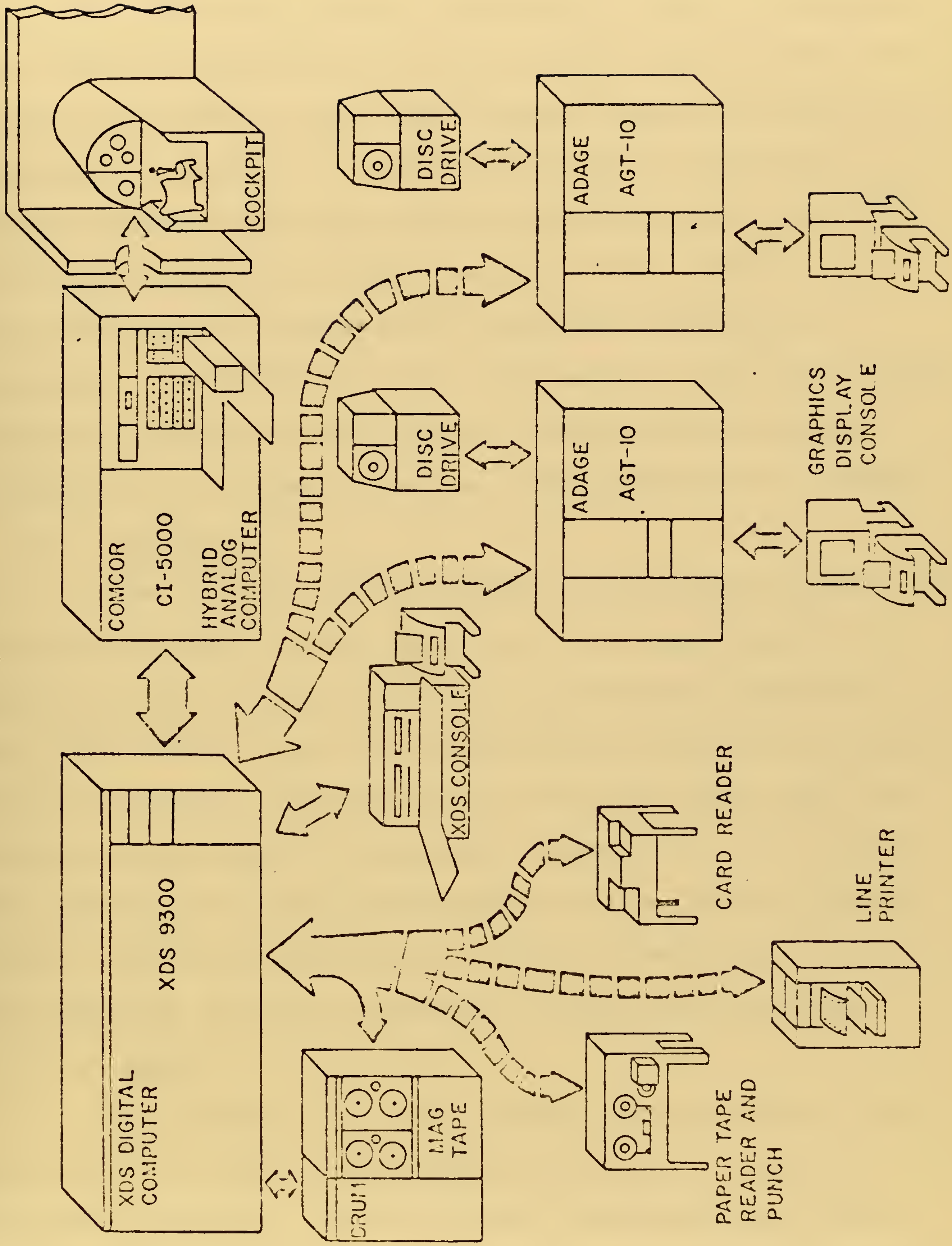

Figure 1. Electrical Engineering Computer Laboratory

printer, paper tape punch, teletypewriter, or magnetic tape drive. The operations and organization of this system allow real-time inputs via special input ports (some of these ports are used to communicate with the CI 5000). This real-time capability allows the XDS 9300 computer system to handle, in addition to the obvious computational applications of a digital computer system, many other applications. Such applications include the real-time monitoring, control, or data reduction of concurrently operating scientific experiments. Furthermore, these applications could also include graphical display on the AGT/10. The XDS 9300 may be programmed in its assembly language (META-SYMBOL) or in FORTRAN.

2. CI 5000

The CI 5000 computer system is an analog computer with a full complement of digital logic. The analog portion is primarily used for the solution of differential equations with the digital logic allowing decision making and switching to be performed. This system, in the stand-along mode, may solve a large number of problems in simulation. The use of the XDS 9300 and the CI 5000 together allow hybrid computations. The major application of hybrid computation is in the modeling of dynamic systems or optimization problems.

3. AGT/10

Each AGT/10 is a small general purpose digital computer with 8K of main (core) memory and magnetic disks for secondary storage. Input is from a teletypewriter, paper tape reader, lightpen, "joystick" type potentiometer, function switches and variable control dial potentiometers. Output

appears on the cathode ray tube, paper tape punch, or teletypewriter. These I/O components, along with some special graphics hardware, make them very effective graphics computers. Their limited storage and lack of a high level language limits their stand-alone capability somewhat. However, when the AGTs are used in conjunction with the XDS 9300, a large number of interactive, graphical problems may be handled.

B. FORTRAN GRAPHICS PACKAGE

The XDS 9300 FORTRAN user presently has access to a number of library subroutines for use in outputting graphics information to the AGT/10 and GATED. The user is primarily concerned with the areas of graphics initialization, output, output with response, and input. Reference [6] contains detailed explanations of the graphics subroutines as well as examples of their use.

1. Graphics Initialization Subroutine--DGINIT

This routine must be called before any of the other graphics subroutines may be called. This subroutine specifies a FORTRAN integer array as the graphics directory, causes GATED to initialize its graphics directory in the AGT/10, and deletes all graphics blocks. This, of course, causes all graphics displays to be erased from the screen. A typical call would be

```
CALL DGINIT (IDEV, IGDIR, NWDIR, IER)
```

IDEV is an integer variable which can assume the values 1 or

2, specifying the number of the AGT being used. IER is an error flag used by all graphics subroutines. Specific values of IER denote various error conditions which are listed in [6]. IGDIR is the first word address of an integer array to be used as the graphics directory. NWDIR is the dimension of the graphics directory; it must be one greater than the number of graphics blocks in existence at any one time. DGINIT zeros out the directory, whose words are subsequently filled by other system subroutines. Each word in the directory corresponds to a single graphics block. When a new block is created, the first non-zero word in the directory is filled with data for that block. As the blocks are created, they are numbered consecutively in ascending order.

2. Graphics Output Subroutine - GRAPHO

This routine outputs the graphics data block (image array) to the AGT/10 and GATED for presentation on the cathrode ray tube. No editing may be performed at the graphics console on blocks which have been output by GRAPHO. A typical call would be of the form:

```
CALL GRAPHO (IDEV, IMAGE, NWORD, IBLK, IER)
```

IDEV specifies to which AGT/10 the block is being passed. IMAGE is the first word address of the graphics data block being output. NWORD is the number of words in that block, and IBLK is the block number (as referred to in the section above). Graphics data blocks contain two types of words: The first (header) word, and as many data words as are needed to specify the figure which is to be drawn. Figure 2(a)

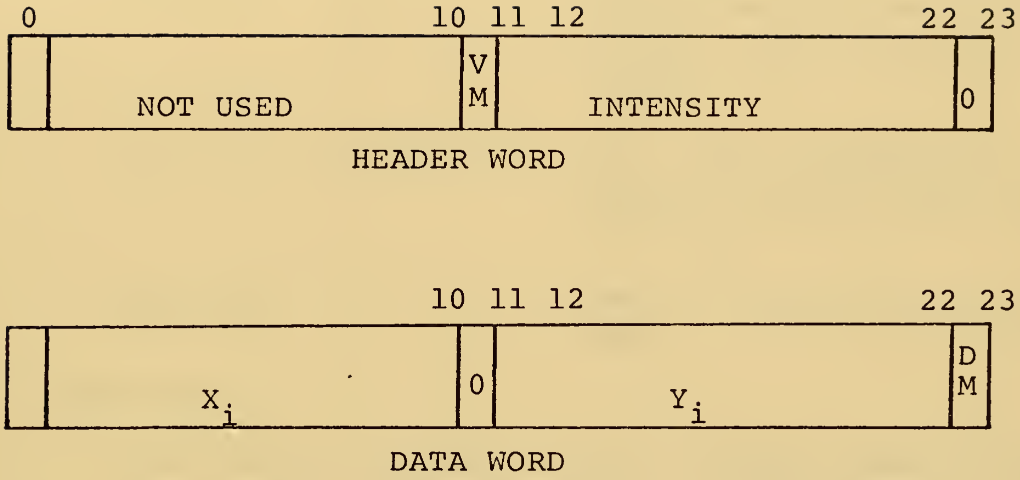

Figure 2(a). Present XDS 9300 Data Structure

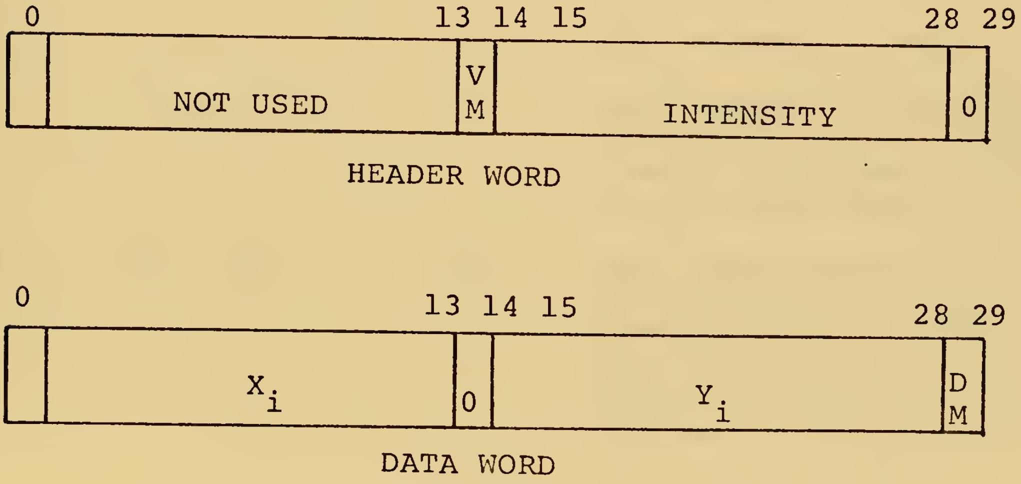

Figure 2(b). Present AGT/10 Data Structure

depicts the format of each of these words. The fields have the following interpretation:

HEADER WORD

bit 11:	VM...	If VM=0 the lines of the data block will be solid. If VM=1 the lines will be dashed
bits 12-22	INTENSITY...	The value in this field is a two's complement number ranging from-1024 to +1023.
bit 23:		Set to 0 and not typically used.

DATA WORD

Each data word has the same format, which allows one word to define a point in the Cartesian coordinate system. Each word is interpreted as the endpoint of a line whose starting point is the previous word.

bits 0-10:	X_i -	This is the X coordinate of the i^{th} point. It is a two's complement number ranging from -1024 to +1023.
bit 11:		Must be set to zero
Bits 12-22:	Y_i -	This is the Y coordinate of the i^{th} point. It is a two's complement number in the same range as X_i.
bit 23:	DM -	When DM=1 the line defined by this word is drawn. When DM=0 the line is a move (not visible, but still exists). This allows the drawing of disconnected images.
	Note:	The maximum values of X_i and Y_i define a 20" square.[1] The screen is a 12" square and the linear portion is a 10" square. Thus, X_i and Y_i must be between -512 and +511 if the figure is to be drawn inside the 10" square.

3. Graphic Output with Response Subroutine-GRAPHR

This subroutine is called with exactly the same parameters as GRAPHO. In addition to passing the image array for display on the cathode ray tube, GRAPHR also causes GATED to enter an edit mode for the block concerned. The block may then be edited at the AGT/10 display. The FORTRAN programmer would normally cause his program to loop until editing operations were complete.

```
CALL GRAPHR (IDEV, IMAGE, NWORD, IBLK, IER)
```

4. Graphics Input Subroutine - GRAPHI

This subroutine inputs an image array from the AGT/10 to the XDS 9300.

```
CALL GRAPHI (IDEV, IMAGE, IBLK, IER)
```

IDEV, IBLK and IER are as defined above. IMAGE is the first word address of an integer array where the data block is to be placed.

5. Special Routines

To construct a graphic image requires a great deal of bit processing. To allow the user to avoid these difficulties, the functions IHEAD and IPACK were created. In addition there is a special subroutine, UNPACK, which extracts values from the fields of the image array words for use as FORTRAN variables.

a. Function to Create Header Word - IHEAD

```
IMAGE (1) = IHEAD (IDSH, INT)
```

In this example, IMAGE (1) is the header word

of an image array. IDSH and INT are variables which IHEAD will place in the VM and INTENSITY fields, respectively (see Figure 2(a)).

b. Function to Create Data Word - IPACK

IMAGE (I) = IPACK (X, Y, IDM)

In this example, IMAGE (I) is the i^{th} word in a graphics data block. X, Y, and IDM are variables which IPACK will place in the X_i, Y_i, and DM fields, respectively.

c. Subroutine to Extract Values from the Image - UNPACK

CALL UNPACK (IMAGE (I), X, Y, IDM)

In this case, IMAGE (I) is the i^{th} word of a graphics data block. X, Y, and IDM are variables where UNPACK will place the values from the X_i, Y_i, and DM fields, respectively. In other words, this subroutine performs the reverse process of IPACK.

C. GATED, EXISTING GRAPHICS AND TEXT EDITOR AT THE NAVAL POSTGRADUATE SCHOOL.

1. Philosophy of GATED

The existing graphics and text editor program, GATED, resides as a permanent file on the system disk packs. It essentially performs three tasks [6]:

(1) Refreshing the cathode ray tube display. The screen is refreshed as part of the program's major cycle at about 40 frames per second, except during data transfers to or from the XDS 9300. During these data transfers, no refresh is performed.

(2) Communicating with the graphics console operator. GATED uses the CRT display, teletypewriter, lightpen, and function switches to communicate with the user. The user can create text and graphics data or edit existing data. For simplicity one can think of creating data as editing an empty data block. During editing of data, the screen will be refreshed, but data will not be transferred between the AGT/10 and the XDS 9300.

(3) Communicating with the FORTRAN graphics package in the XDS 9300 (described in III B. above).

GATED divides its data into two sets, graphics and text. This thesis has concerned itself only with graphics data, as it has been in that area that the design improvements have been made. The graphics data blocks, as described in III B. above and Figure 2(a), are transferred via the interface to the AGT/10. GATED displays all of the data blocks that is has stored in the AGT/10 memory. The FORTRAN graphics package can only send or retrieve one block of data at one time. The user can edit (or create) only one block of data at a time. To edit a block, the following steps must be performed in the prescribed order [6]:

1. Specify the user's desire to edit data.

2. Specify which block is to be edited or that a new block is to be created. GATED creates an empty block in the latter case.
3. Enter the edit mode.
4. Edit the block.
5. Terminate the edit mode.

Most of the above steps are accomplished by pushing the proper function switches. This is worthy of mention in that it will be shown that RADIK, while preserving the basic editing philosophy of GATED, allows more flexibility in the use of the function switches and other hardware devices (i.e., joystick and variable control dials).

2. Data Structure of GATED

GATED performs the necessary shifting and masking required to change the 24 bit XDS 9300 data word into a 30 bit AGT/10 data word (see Figure 2(b)). A zero in bit 14 of the header word will produce a picture with solid lines, while a one in bit 14 will produce a dashed figure. Bits 15-29 are used to specify the amount of intensity with which the block is to be displayed. Bits 1-13 are unused.

In the data words, bits 1-13 are used for the X coordinate for the endpoint of a line, while bits 15-28 are used for the Y coordinate of that point. A zero in bit 29 will result in a move to that point, whereas a one in bit 29 will produce a draw. Bit 14 is unused and always set to zero.

There is a subroutine in GATED(SHRNK), which is used to recover unused storage from graphics data blocks. SHRNK

searches the graphics data block for two or more consecutive moves and deletes all but the last move of the set of consecutive moves. Since moves are not displayed, no effect is seen on the screen display, while the memory words previously occupied by the moves are freed for further utilization. As will be seen later, this subroutine had to be modified for inclusion in RADIK in order to accommodate a new graphics block data structure.

3. Control of GATED

The user utilizes the function switches, teletypewriter and lightpen to provide input to GATED. The function switches provide controls for branching to various modes and subroutines which will perform the necessary steps for creating new data blocks or referencing existing data blocks and allowing them to be edited. An overlay labeled "GATED", when placed over the function switches, will indicate what action will be performed by depressing a particular switch (see Figure 3). The upper left hand function switch is designated FNS1 and the lower right hand switch is designated as FNS16. [6]

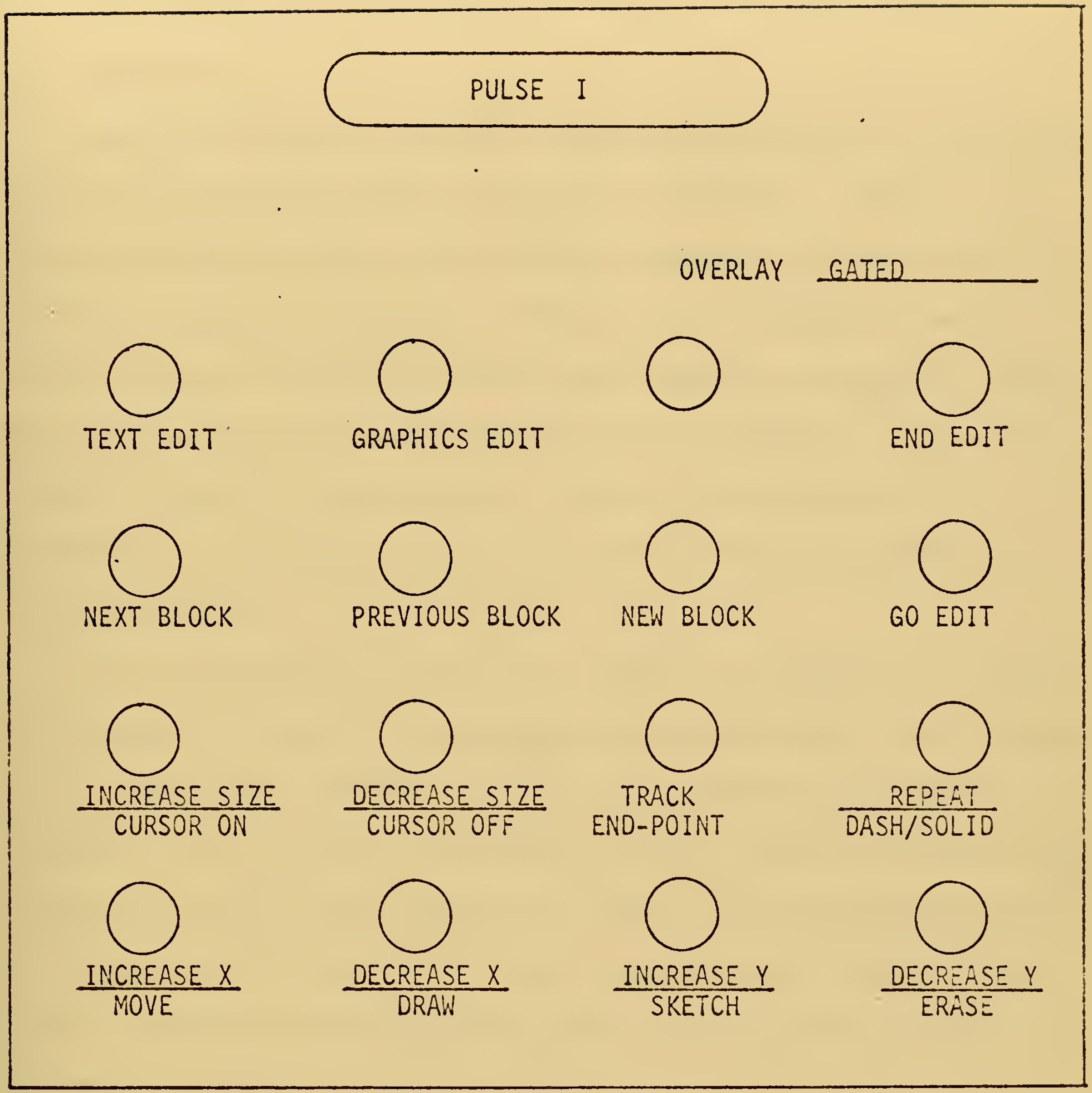

Figure 3
Function Switches
(GATED Overlay Shown)

IV. RADIK, AN INTERACTIVE GRAPHICS AND TEXT EDITOR

A. OBJECTIVES

GATED provides for AGT/10 stand-alone graphics operations as well as XDS 9300-AGT/10 graphics displays. However, it is felt that, in both modes, GATED fails to fully utilize existing graphics hardware. RADIK is an attempt to more fully utilize available interactive graphics hardware, using the philosophy and program structure of GATED as a base from which to build. Specifically, the following goals were proposed as reasonable objectives for the work presented in this thesis:

(1) The display hardware is capable of receiving varying inputs for use in the setting of scale factor, DX (change in X coordinate) and DY (change in Y coordinate). GATED, however, fails to take advantage of this fact and sends values which always set scale factor to unity and DX and DY to zero. It was desired, then, to be able to pass scale factor and DX/DY information from the XDS 9300 to the AGT/10 as part of a graphics block.

(2) To utilize the function switches for editing of graphics displays, both in stand-alone and XDS 9300 modes of operation. GATED presently limits the use of the function switches to those operations denoted in Figure 3 and described in [6].

(3) To utilize the settings of the function switches, variable control dial potentiometers and joystick potentiometers

as control information for the XDS 9300 FORTRAN programs. GATED, although it samples the function switches, does not interrogate the variable control dials or joystick, and furthermore, does not have the capability to transmit any of the above listed information to the XDS 9300.

(4) To make the necessary changes in the XDS 9300 monitor and FORTRAN graphics package to accommodate objectives (1), (2) and (3). This would require modification of the data structures themselves, as well as writing FORTRAN callable assembly language subroutines to process the desired information.

The objectives stated above indicate that the work presented in this thesis encompassed both the AGT/10 and XDS 9300 computers. As mentioned in III above, these computers differ significantly in their design philosophy and operating characteristics. Since the work required an intimate knowledge of the internal structures of each machine, it was decided that it could best be performed by two people. Accordingly, the author is responsible for the AGT/10 software discussed herein, and Lcdr Ralph H. Stowell, Jr., USN, is responsible for the work performed on the XDS 9300. Close communication was required throughout the design process, however, in order to be able to accommodate alterations on both sides of the system and to establish the required interface. The major portion of this work, then, consisted of assembly language (ADEPT) programming on the AGT/10 and assembly language (META-SYMBOL) programming on the XDS 9300. FORTRAN programs were used to test and demonstrate the work.

B. RESULTS

1. Data Structures

a. XDS 9300

Perhaps the single most important concept upon which this work relies is the change in the XDS 9300 data structure for graphics blocks. In order to transmit scale factor, DX, and DY as inputs to the AGT/10 graphics and text editor, a radical change was needed in the configuration of the image array. The result is that scale factor information has been placed in the previously unused portion of the header word, while DX and DY are placed in the second word of the array, as illustrated in Figure 4(a). Remaining data words of the block are left unchanged. While this requires one more word of storage for a given block, the advantages gained make this cost trivial.

Previously, if the user had desired to change scale factor or DX/DY, it could never actually be accomplished. Such changes could be simulated, however, by running each of the data words through a FORTRAN "DO" loop to change the X and Y coordinates in such a manner as to effectively change the size or displacement of the figure being displayed. With the new block structure, only the word with the desired information need be modified (i.e., header word or second word). This saves execution time (one assignment statement versus the "n" assignment statements and conditional branch required for a "DO" loop), and also, retains the original coordinate information of the array.

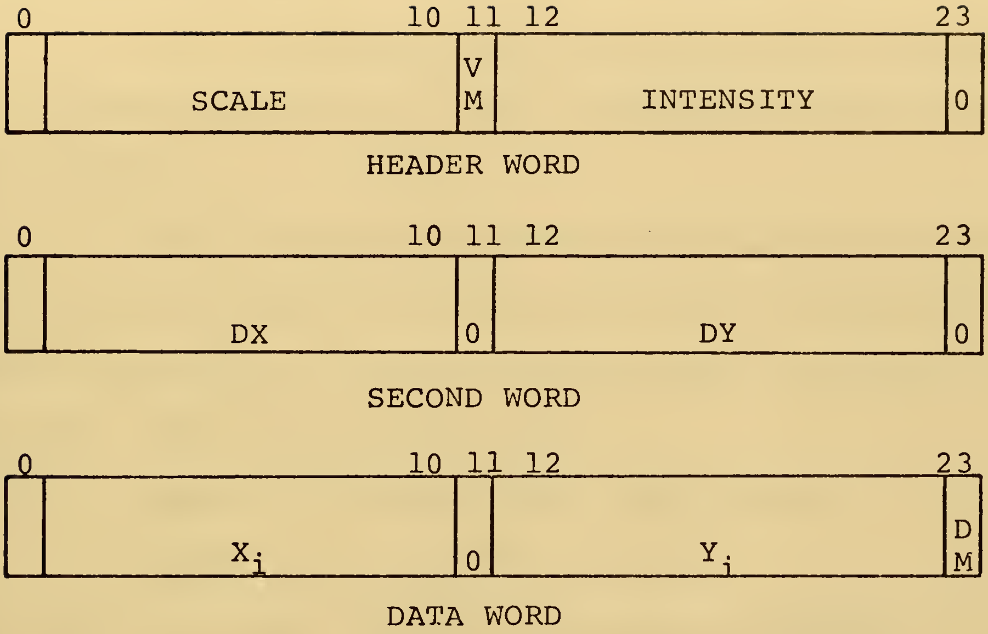

Figure 4(a). Proposed XDS 9300 Data Structure

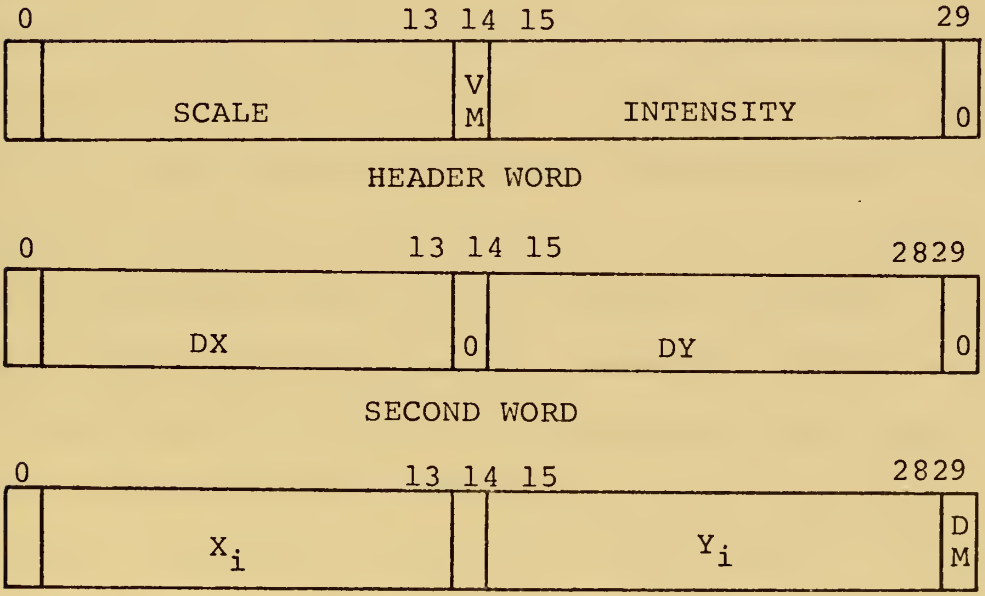

Figure 4(b). Proposed AGT/10 Data Structure

To implement such a data structure, new routines were required to build the graphics data block's first two words. This was accomplished in the following manner:

(1) The header word was built using a specially written FORTRAN subroutine, IPAKM, which is listed in COMPUTER PROGRAM 1.

(2) The second word was built using the function IPACK (described above) with parameters DX, DY and zero. FORTRAN statements for building these first two words of an image array take the following form:

```
IMAGE (1) = IPAKM (SKAL, IDSH, INT)
IMAGE (2) = IPACK (DX, DY, 0)
```

Where SKAL is an octal number in the range $0000_8 \leq SKAL \leq 1776_8$. The choice of this range of numbers is significant and deserves further comment. Since 11 bits (bits 0-10) are used for SKAL, the largest positive number that could be represented is 3776_8. The hardware of the AGT/10 will, in fact, translate 3776_8 into a scale factor of unity. However, since one of the desired functions of RADIK was the ability to magnify a figure as well as scale it down, 1776_8 was chosen to represent unity to the FORTRAN programmer. In this way, the user will be able to enlarge a figure to twice its original size. It should be pointed out that only 11 bits are used for scale factor information; therefore, no odd numbers can be used, since this might result in setting bit 11 (and therefore IDSH) to 1, even though a solid figure is desired. IDSH, and INT are integer variables as described previously. DX and DY are values in the same range as for

X-Y coordinates. The parameter 0 is passed to IPACK for the second word, since it might otherwise be interpreted as a draw. No changes were necessary for constructing the remaining words of the data block, and either GRAPHO or GRAPHR could still be used to output the block to the AGT/10.

b. AGT/10-RADIK

Since the new XDS 9300 data structures are the most important single concept upon which this work relies, they are responsible for much time and effort to facilitate their use. Since the program structure of GATED was being used as a base, the new block structure being transmitted by the XDS 9300 necessitated extensive program changes in order to be accommodated by the graphics and text editor. The physical size of the program (80 pages of listing) dictated many detailed alterations, mostly in the area of indexing and indirect addressing. Other factors dealing with the data structure also had to be considered; whereas GATED had always set scale factor to unity and DX/DY to zero, RADIK checks the first and second words of an image array for that information and transfers it to the appropriate hardware for implementation. The stand-alone mode of the AGT/10 had to be considered as a separate problem, since the AGT/10 creates empty blocks in that mode. In addition to the same indexing and indirect addressing problems as were encountered in the XDS 9300-AGT/10 mode, it was required to set scale factor initially to unity and DX/DY to zero. In addition, since the function switches were to be used for performing new routines

in both modes, variables which were common to both modes had to be used when implementing that portion of the work.

2. Routines

Several new routines were written in order to allow the user to have more flexibility and versatility in FORTRAN programming and to utilize available hardware control devices in both stand-alone and XDS 9300-AGT/10 operation.

a. AGT/10 - RADIK routines.

The portion of the work presented here consisted of implementing several function switches to perform graphics editing functions while RADIK is in a graphics wait mode. Previously, the switches were not used for functions other than those depicted in Figure 3. The switches and their new functions are listed below:

FNS1 - translates figure to the left
FNS2 - translates figure to the right
FNS3 - scales figure down
FNS5 - translates figure upward
FNS6 - translates figure down
FNS7 - magnifies figure
FNS8 - rotates the figure

In order for a user to edit a graphics block he must perform the following steps in order [6]:

1. Select the graphics edit mode.

Depress GRAPHICS EDIT (FNS2). The message "GRAPHIC BLOCK SELECT MODE BLOCK1" will appear on the lower edge of the screen if operating in conjunction with the XDS

9300. Otherwise the message "GRAPHIC EDIT REQUEST" appears at the bottom of the screen.

2. Select the graphics block to be edited.

Depress NEXT BLOCK (FNS5) once for each block to be skipped. If the desired block has been passed, depressing PREVIOUS BLOCK (FNS6) will back up one block at a time until the desired block is reached.

At this point, it is also possible to create a new block. To do this, depress NEW BLOCK (FNS7). RADIK allocates all of free core to the new graphics block. This new block becomes the next sequential block. When edit is terminated, the new block will be shrunk to the number of words actually used, thus allowing the unused words to be returned to free core.

3. Start the edit mode.

Depress GO EDIT (FNS8) to enable the graphic edit mode for the selected block. The following message will appear at the bottom of the screen---"GRAPHIC EDIT MODE BLOCK" followed by the block number. If a block has been output from the XDS 9300 via a GRAPHR, these first 3 steps are performed for the user by RADIK.

4. Graphics editing operations.

It is at this point that the function switches listed above are enabled for their new editing capabilities. Figure 5 depicts an overlay for the function switches for use with RADIK. All of the new functions will be performed continuously if the associated switch is held down. Depressing FNS4 will end the editing process (as in GATED) and the block

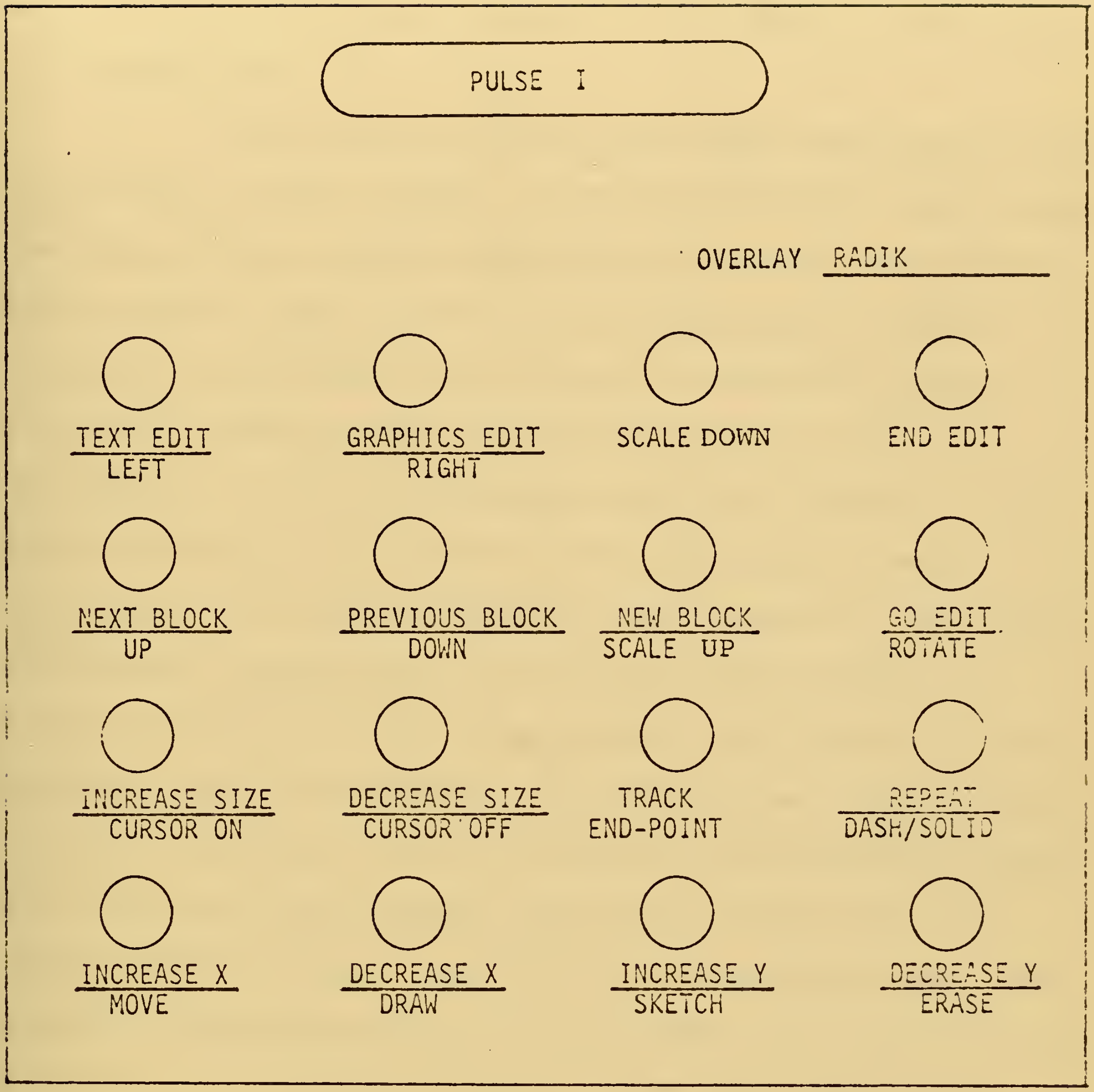

Figure 5
Function Switches
(RADIK Overlay Shown)

will contain the information associated with the figure's present perspective and position on the screen. If operating with the XDS 9300, this information will then be transmitted to the XDS 9300.

An interesting difficulty arose during this portion of the work. While editing, if FNS8 is held down long enough, one will notice a gradual diminishing in size of the figure being edited. This is due to the accumulation of small round-off errors which are inherent in the AGT/10 multiplication process. While it appears that the figure is diminishing in scale, in actuality the scale factor is not being changed at all; however, each data word's X and Y coordinates are being altered by minute amounts. While this is obviously undesirable, the only solution would be to limit the number of words in a graphics data block so that a temporary block of arbitrary size could be used to store the original coordinate data and counteract the round-off errors. This would not only restrict the user, but would also increase RADIK's storage requirements and execution time. The proposed corrective action is to use FNS8 for a short period of time (not a severe restriction, since most users would desire less than 360° of rotation for a change in perspective), and then correct the size by using FNS7 to magnify the figure.

The structure of the data words themselves also gave rise to a minor difficulty during the implementation of the new editing functions. It was noticed that, when translating a figure horizontally using FNS1 and FNS2, there was a very slight change in the figure's vertical displacement.

This was due to the method being employed to effect the horizontal translation. The method consisted of adding (or subtracting) a small increment to the DX field of the second word in the graphics block whenever FNS1 (or FNS2) was depressed. However, since only half of the second word is being used for the DX field, the addition of this increment would sometimes cause an "overflow" into that portion of the word which is used for DY. This was solved by extracting the information from the DX field, incrementing it, and then inserting it back into the second word. In this way, the DY field will not be inadvertently changed.

RADIK also had to be modified due to the addition of three new routines, VCD, JOYSTK, and SWITCH, to the XDS 9300 FORTRAN graphics package. As their names imply, these routines are used to obtain information from the variable control dials (VCD), joystick (JOYSTK) and function switches (SWITCH). These routines were written as part of the work for this thesis and are described in IV. 2b. below. The writing of these routines necessitated modifying RADIK to:

(1) Receive the new routines' request for information;

(2) Obtain the desired information; and

(3) Transmit the information to the XDS 9300.

Obtaining the desired information (function switch states, variable control dials and joystick potentiometer values) was not a problem. In fact, procedures already existed to procure the states of all the above listed devices. What was needed, then, was a way to communicate the information

request and reply between the XDS 9300 and the AGT/10. This was accomplished by altering the XDS 9300 monitor to handle the new routines in a manner similar to that used for other library subroutines. In addition, communications routines were built into RADIK to process the new routines in a manner similar to GRAPHO and GRAPHR.

b. XDS 9300 Subroutines

Three subroutines were written to allow the XDS 9300 user to request and receive the status of the variable control dials (VCD), joystick (JOYSTK), and function switches (SWITCH). VCD and JOYSTK are written in META-SYMBOL, but are FORTRAN callable. SWITCH is written in FORTRAN. All are included in Computer Program 1. The META-SYMBOL routines have been incorporated into the XDS 9300 subroutine library and have been fully tested in a number of programs.

1. VCD

VCD is the assembly language, FORTRAN callable subroutine which requests and receives the readings of all six variable control dial potentiometers. A typical call would be of the form:

```
CALL VCD (IDEV, IDIAL, DIAL, IER)
```

IDEV is the number of the AGT being used. IDIAL is the first word address of an integer array of dimension six, and DIAL is the first word address of a real array of dimension six. IER is the error flag used in all graphics subroutines. As a result of a call to VCD, values of the six variable control dial potentiometers are placed in DIAL (1) through DIAL (6).

2. JOYSTK

JOYSTK is the assembly language, FORTRAN callable subroutine which requests and receives the readings of all three potentiometers which are attached to the joystick. A typical call would be:

CALL JOYSTK (IDEV, JOY, AJOY, IER)

IDEV is the number of the AGT being used. JOY is the first word address of an integer array of dimension three, and AJOY is the first word address of a real array of dimension three. IER is the error flag used in all graphics subroutines. As a result of a call to JOYSTK, the values of the three joystick potentiometers are placed in AJOY (1) through AJOY (3).

Note that both VCD and JOYSTK are called with what appears to be an "extra" parameter, namely the integer arrays IDIAL and JOY. This is due to the inherent nature of the communications link between RADIK and the XDS 9300. The analog values of the potentiometers are digitized and transmitted as single words to the XDS 9300. Since an integer variable occupies one word of storage in the XDS 9300, these digitized values are first placed in the integer arrays. A specially written subroutine, XFORM, is then called from within VCD and JOYSTK. XFORM transforms the single word digitized quantities into real variables and places them in the double word real arrays.

3. SWITCH

SWITCH is a FORTRAN subroutine which requests and receives the status of the 16 function switches. All 16 switches are interrogated to see whether or not they are set,

and this information is passed to the XDS 9300 in the lower 16 bits of a single word. A logical function, CHANGE, is then used to see whether a particular switch is set or not. Typical calls to SWITCH and CHANGE would be:

```
CALL SWITCH (IDEV, ISWITCH, IER)
IF (CHANGE 3, SWITCH)...
```

In this example, IDEV and IER are the AGT number and error flag respectively. ISWITCH is the integer variable used to store the information and 3 is the number of the switch whose status is being requested. SWITCH and CHANGE, then, provide a means for using the setting of the function switches as branching controls for FORTRAN programs.

IV. APPLICATIONS

The results discussed in Section III allow the user much more flexibility in utilizing the available equipment. Translation, rotation, and scaling may be performed in conjunction with the XDS 9300 or in AGT/10 stand-alone mode. One immediately obvious advantage over GATED is that the user may now use GRAPHR to transmit a graphics block from the XDS 9300 whose data words are empty, draw the basic picture on the screen with the lightpen, and then translate, rotate, and scale the figure to his specifications before sending the information back to the XDS 9300. Previously, the user would have had to make computations and measurements before writing his program to obtain the desired picture. Even then, a figure might not have appeared exactly as desired, so that additional time consuming measurements and computations had to be made. Similarly, the user may now arrange several figures on the screen in their desired relationships to one another by use of the function switches to achieve the proper configuration in an on-line, interactive manner as opposed to lengthy "hit and miss" techniques.

In addition to their editing capabilities, the function switches, along with the variable control dials and joystick, can now be used as controls for the branching of FORTRAN programs. Of course, they may also be used to directly change a display by assigning the values to a graphics data block word.

Applications of RADIK and the new FORTRAN graphics package will be limited only by the imagination of the programmer. No applications have been lost from the existing system and the new data structures and available hardware open up countless new possibilities.

V. CONCLUSIONS

A generalized graphics display system, one which would allow a user with a specialized visual display requirement to produce the display with relative ease, would have to meet such general criteria as the following [2]:

(1) It must be on-line or at least closely appear to be so. This tends to reduce trial-and-error frustrations to a minimum.

(2) It must be conversational, in that every user request is met with a response from the computer. This is important in that certain operations may not have immediately obvious results.

(3) It must be versatile by offering facilities both for constructing pictures out of primitive geometries and symbolic figures, and for performing transformations upon such pictures.

(4) It must offer continuity between constructs by having facility for dynamic storage of partial or complete pictures, both internal and external to the computer. A user should be able to store, retrieve, and superimpose previously drawn pictures.

It is felt that the graphics display system at the Naval Postgraduate School Computer Laboratory will meet these criteria, if RADIK is implemented as the graphics and text editor and the required associated changes are made in the XDS 9300 monitor system and subroutine package.

If implemented, the largest contribution that the work presented in this thesis will make to the graphics display system at the Naval Postgraduate School is the increased versatility and flexibility of FORTRAN programs in controlling a display on the AGT/10. This has been accomplished simply by utilizing more fully the available interactive hardware configurations.

```
      INTEGER Z
      INTEGER FRAME
      INTEGER SCALE
      LOGICAL CHANGE
      DIMENSION ISWITCH(1)
      DIMENSION ITDIR(2), IGDIR(5)
      DIMENSION FRAME(25)
      DIMENSION X(25), Y(25), IMD(25)
      DIMENSION DIAL(6), JOY(3)
      DIMENSION IDIAL(6)
      DIMENSION AJOY(3)
      NAMELIST IDEV,IER
      IDEV=2
      OUTPUT(101) 'TYPE IDEV=1* AND A CARRIAGE RET IF AGT1 TO BE USED'
      INPUT(101)
      CALL DGINIT(IDEV,IGDIR,5,IER)
      IF(IER.NE.0) OUTPUT(101) IER, '   DGINIT'
      CALL DTINIT(IDEV,ITDIR,2,IER)
      IF(IER.NE.0) OUTPUT(101) IER, '   DTINIT'
      DX=0
      DY=0
      SCALE=7760000B
      FRAME(1)=IPAKM(0,10,SCALE)
      FRAME(2)=IPACK(DX,DY,0)
      DO 50 K=3,25
      FRAME(K)=IPACK(0,0,0)
   50 CONTINUE
      CALL GRAPHR(IDEV,FRAME,25,1,IER)
      IF(IER.NE.0) OUTPUT(101) IER, '   GBLK1'
    1 IF(MOD(IGDIR(1),8).EQ.0) GO TO 1
      CALL GRAPHI(IDEV,FRAME,1,IER)
      IF (IER.NE.0) OUTPUT(101) IER, '   IGBLK'
      CALL UNPAKM(FRAME(1),INT,IDSH,SCALE)
      CALL UNPACK(FRAME(2),X(1),Y(1),IMD(1))
      CALL DGINIT(IDEV,IGDIR,5,IER)
```

```
      DX=X(1)
      DY=Y(1)
      FRAME(1)=IPAKM(IDSH,INT,SCALE)
      FRAME(2)=IPACK(DX,DY,0)
      CALL GRAPH8(IDEV,FRAME,25,1,IER)
      N=100000
      CALL DELAY
      CALL DGINIT (IDEV,IGDIR,5,IER)
      CALL GRAPHR (IDEV,FRAME,25,1,IER)
      DX=-.5
      DY=-.5
      CALL GRAPH8(IDEV,FRAME,25,1,IER)
      DO 100 I=1,15
      DX=DX+.1
      DY=DY+.1
      FRAME(2)=IPACK(DX,DY,0)
      CALL GRAPH8(IDEV,FRAME,25,1,IER)
      IF (IER.NE.0) OUTPUT(101) IER, '  GBLK1A'
      N=100000
      CALL DELAY
  100 CONTINUE
      CALL DGINIT(IDEV,IGDIR,5,IER)
      IF(IER.NE.0) OUTPUT(101) IER, '   DGINIT'
      DX=0
      DY=0
      SCALE=00000000B
      FRAME(1)=IPAKM(0,10,SCALE)
      FRAME(2)=IPACK(DX,DY,0)
      CALL GRAPH8(IDEV,FRAME,25,1,IER)
      IF(IER.NE.0) OUTPUT(101) IER, '   GBLK1'
      N=100000
      CALL DELAY
  200 SCALE=SCALE+00100000B
      CALL VCD(IDEV,IDIAL,DIAL,IER)
      DX=DIAL(1)
```

```
DY=DIAL(2)
FRAME(1)=IPAKM(0,10,SCALE)
FRAME(2)=IPACK(DX,DY,0)
CALL JØYSTK(IDEV,JØY,AJØY,IER)
IF(AJØY(1).GT..25) STØP
CALL GRAPHØ(IDEV,FRAME,25,1,IER)
IF (IER.NE.0) ØUTPUT(101) IER, '  GBLK1A'
N=100000
CALL DELAY
ØUTPUT(101) 'LØØPING'
CALL SWITCH(IDEV,ISWITCH,IER)
IF(IER.NE.0) ØUTPUT(101) IER,'HEREWEGØ'
ØUTPUT(101) 'ATXFØRM'
IF(CHANGE(3,ISWITCH)) STØP
ØUTPUT(101) 'ATVCD'
IF(SCALE.LT.37760000B) GØ TØ 200
CALL DGINIT(IDEV,IGDIR,5,IER)
STØP
END
```

```
      FUNCTION IPAKM(IDSH,INT,SCALE)
      IPAKM=LIOR(IHEAD(IDSH,INT), SCALE)
  100 RETURN
      END
```

```
SUBROUTINE UNPAKM(IMAGE(1),INT,IDSH,SCALE)
INTEGER TEMP
INTEGER SCALE
IDSH=0
INT=10
TEMP=LAND(IMAGE(1),00010000B)
SCALE=LAND(IMAGE(1),77760000B)
IF(TEMP.NE.00000000B) IDSH=1
RETURN
END
```

```
      LOGICAL FUNCTION CHANGE(Z,ISWITCH)
      INTEGER Z
      CHANGE=.FALSE.
      NO=16-Z
      ITEMP=LCRS(ISWITCH,NO)
      IF(LAND(ITEMP,00000001B).EQ.0) GO TO 50
      CHANGE=.TRUE.
   50 RETURN
      END
```

```
$VCD     PZE     0
         BRM     9SETUPN
         PZE     4
         PZE     0
         PZE     0
         PZE     0
         PZE     0
         LDA     *VCD+3
         BRM     DVNØCK               IS DEV CØRRECT
         BRU     VCDDER               NØ
         STA     VCDCL+2              STØRE IDEV FØR USE BY D/EXEC
         STA     FIØBFL               YES, SET NEW I/Ø FLAG BUSY
         LDA     VCD+4
         STA     VCDXFM+2
         LDA     VCD+5
         STA     VCDXFM+3
         LDA     =5
         STA     VCDXFM+4
         LDA     =01000000
         STA     VCDSWØ               STØRE CØNTRØL NØ. IN SWØ1
         LDA     VCD+4
         STA     VCDSWØ+1             STØRE FWA IN SWØ2
         SUB     =017777              CHECK FWA <= 20,000
         SKG     =0
         BRU     VCDMER               GØ SET ERRØR
         EØM     032004               DISABLE BØTH AGT'S
VCDCL    BRM     D\EXEC               GRAPHIC ØUTPUT CALL
         PZE     2
         PZE     0
         PZE     VCDSWØ               SWØ FWA
         EØM     032001               ENABLE AGT1
         EØM     032002               ENABLE AGT2
         LDB     =0
         LDA     *VCD+3               DEV NØ.
         SKA     IØBFL                LØØP UNTIL I/Ø NØT BUSY
```

```
        BRU     $-2
        CØPY    (4,5)               SET ERRØR FLAG
        BRU     VCDER
VCDDER  LDA     =1                  BAD DEV NØ.
VCDER   STA     *VCD+6
        SKU     =0
        BRU     VCDXFM
        BRR     VCD                 RETURN
VCDMER  LDA     =10                 MEMØRY NØT ACCESSIBLE BY INTERFACE
        BRU     VCDER               ERRØR EXIT
VCDXFM  BRM     XFØRM
        PZE     3
        PZE     0
        PZE     0
        PZE     0
        BRR     VCD
VCDSWØ  PZE     0                   ALLØCATE SWØ
        PZE     0
        PZE     0
        END
```

```
$XFORM  PZE    0
        BRM    9SETUPN
        PZE    3
IDUMMY  PZE    0
DUMMY   PZE    0
        PZE    0
        LDA    =0
        STA    QQ
        LDA    =1
        STA    I
        LDA     XFORM+5
        STA     1PTMO
XX      LDA    *IDUMMY
        LDB    =0
        FLA    QQ
        STD    *DUMMY
        MPO    IDUMMY
        MPT    DUMMY
        MPO    I
        SKR    1PTMO
        BRU    XX
        BRR    XFORM
1PTMO   RES    1
QQ      RES     1
I       RES     1
        END
```

```
$JOYSTK PZE     0
        BRM     9SETUPN
        PZE     4
        PZE     0                   DEV NO
        PZE     0                   FIRST WORD ADDRESS
        PZE     0
        PZE     0                   ERR FLAG
        LDA     *JOYSTK+3
        BRM     DVNOCK              IS DEV CORRECT
        BRU     JOYDER              NO
        STA     FIOBFL              YES, SET NEW I/O FLAG BUSY
        STA     JOYCL+2             STORE IDEV FOR USE BY D/EXEC
        LDA     JOYSTK+4
        STA     JOYXFM+2
        LDA     JOYSTK+5
        STA     JOYXFM+3
        LDA     =2
        STA     JOYXFM+4
        LDA     =02000000
        STA     JOYSWO              STORE CONTROL NO. IN SWO1
        LDA     JOYSTK+4
        STA     JOYSWO+1            STORE FWA IN SWO2
        SUB     =017777             CHECK FWA <= 20,000
        SKG     =0
        BRU     JOYMER              GO SET ERROR
        EOM     032004
JOYCL   BRM     D\EXEC              GRAPHIC OUTPUT CALL
        PZE     2
        PZE     0                   DEVICE NO.
        PZE     JOYSWO              SWO FWA
        EOM     032001              ENABLE AGT1
        EOM     032002              ENABLE AGT2
        LDB     =0
        LDA     *JOYSTK+3           DEV NO.
        SKA     IOBFL               LOOP UNTIL I/O BUSY FLAG NOT BUSY
```

```
         BRU     $-2
         COPY    (4,5)                   SET ERROR FLAG
         BRU     JOYER
JOYDER   LDA     =1                      BAD DEVICE NO.
JOYER    STA     *JOYSTK+6
         SKU     =0
         BRU     JOYXFM
         BRR     JOYSTK                  RETURN
JOYMER   LDA     =10                     MEMORY NOT ACCESIBLE BY INTERFACE
         BRU     JOYER                   ERROR EXIT
JOYXFM   BRM     XFORM
         PZE     3
         PZE     0
         PZE     0
         PZE     0
         BRR     JOYSTK
JOYSWO   PZE     0                       ALLOCATE SWO'S
         PZE     0
         PZE     0
         END
```

```
$SWITCH PZE     0
        BRM     9SETUPN
        PZE     3
        PZE     0                 DEV NØ.
        PZE     0                 ADDRESS
        PZE     0                 ERRØR FLAG
        LDA     *SWITCH+3
        BRM     DVNØCK            IS DEV CØRRECT
        BRU     SWDER             NØ
        STA     FIØBFL            YES, SET NEW I/Ø FLAG BUSY
        STA     SWCL+2            STØRE IDEV FØR USE BY D/EXEC
        LDA     =03000000
        STA     SWSWØ             STØRE CØNTRØL NØ IN SWØ1
        LDA     SWITCH+4
        STA     SWSWØ+1           STØRE FWA IN SWØ2
        SUB     =017777           CHECK FWA <= 20,000
        SKG     =0
        BRU     SWMER             GØ SET ERRØR
        EØM     032004            DISABLE BØTH AGT
SWCL    BRM     D\EXEC            GRAPHIC ØUTPUT CALL
        PZE     2
        PZE     0                 DEVICE NØ.
        PZE SWSWØ                 SWØ FWA
        EØM     032001            ENABLE AGT1
        EØM     032002            ENABLE AGT2
        LDB     =0
        LDA     *SWITCH+3         DEV NØ.
        SKA     IØBFL             LØØP UNTIL IØB BUSY NØT BUSY
        BRU     $-2
        CØPY    (4,5)             SET ERRØR FLAG
        BRU     SWER
SWDER   LDA     =1                BAD DEVICE NØ.
SWER    STA     *SWITCH+5
        BRR     SWITCH            RETURN
SWMER   LDA     =10               MEMØRY NØT ACCESSIBLE BY INTERFACE
```

```
        BRU     SWER                ERROR EXIT
SWSWO   PZE     0                   ALLOCATE SWO'S
        PZE     0
        PZE     0
        END
```

```
[ USNPGS GRAPHICS/TEXT EDITOR
[ H.D.K., E.L.B.
[ VERSION 2, REV. 1, 6 JUNE 72
[    MODIFIED FROM ATEXT VERSION 8/11/70
[     MODIFICATIONS BY R.H.S.   R.F.A.

EXPUNGE

TITLE          RADIK
ENTRY          RADIK,GATE1,GATEX

MACRO1         NUL; NULLL ENDM

MACRO1         IFCHANGE       (AA)
               JPSR           IFC                [CALL SUBR.
IREPEAT Q,(AA)
               SWITCHNO       (Q)
ENDI
                              0
ENDM

MACRO1         IFSWITCH       (AA)
               JPSR           IFS
IREPEAT Q,(AA)
               SWITCHNO       (Q)
ENDI
                              0
ENDM

MACRO1         SWITCHNO       (CC,BB)
DECIMAL
```

```
AA = CC
OCTAL
                AAJKJKJKJK + BB
ENDM

MACRO1          P (A1,A2,A3,A4,A5)
A1JBJK + A2JB + A5JBJK + A3JBJK + A4JB + 0
ENDM

MACRO1          ORIGIN
                P (13,106.,11,40.)
                P (10,22,37)
ENDM

ZZZ=101                          [ASCII CODE DEFINITIONS
IREPEAT ZZ,(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,Q,Q,R,S,T,U,V,W,X,Y,Z)
ZZ\. = ZZZ
ZZZ = ZZZ + 1
ENDI
P.=120

ENDCORE = 13777                                    [ADDED CORE FOR DISK SYSTEM
[ PIVOT DEFINITIONS:

CLKPV = 77755;  LPNPV = 77760
EOVPV = 77756;  EOLPV = 77757
LCGAD = 77735;  LCGEA = 77736
LCGEB = 77737;  AROPV = 77771

[EXTRA INSTRUCTION DEFINITIONS:

MD10 = 30000JH;  MD11 = 31000JH;  AR05 = 25100JH
```

```
MD07 = 27000/H;  MD06 = 26000/H;  AR11 = 71100/H
MD05 = 25000/H;  OPIO = 00600/H;  ARBR = 62100/H
ARLS = 17500/H;  ARRS = 17440/H;  DIVI = 16440/H
MPYL = 15420/H
MPYU = 15400/H



[ MAIN ENTRY TO PROGRAM

GATE1:
RADIK:          JUMP            .
                MDAR'I          RADIK
                ARMD            DBUG
                JPLS            .+2
                JPSR            $AXINT
                MDAR            $LD             [SET FREE STORAGE
                ARMD            FREE
                MDAR'L;         ENDCORE
                ARMD            TPTR            [SETUP TEXT ALLOCATION PTR.
                ARMD'O          $OVL1A
                ARMD'O          $OVL2A
                ARMD'O          $OVL3A          [UNALLOCATE AMRMX'S OVERLAY AREAS
                MDAR'L;         TBLK            [INIT. BLOCK LISTS
                ARMD            TBLK
                MDAR'L;         GBLK
                ARMD            GBLK
                ARXO'F
                ARMD            GBLKN           [RESET NO. BLOCKS
                ARMD            TBLKN
                ARMD            CBLKP           [RESET CURRENT POINTER
```

```
                ARMD            CURFG           [RESET CURSAR FLAG
                MDAR'F          CLKS            [SETUP CLOCK SUBR.
                ARMD            CLKPV
                MDAR            C1
                ARMD            MODE

GATEX:          ARXO'F
                ARMD            $GTWD1
                MD10'O'L                        [TURN ON CLOCK
                1000/H
[ START OF MAJOR FRAME LOOP

SA:             MDAR            TOLDS           [SAVE OLD SWITCHES
                ARMD            OLDS
                MDIC'O'H        C40
                S5AR'F'H                        [SAMPLE NEW SWITCHES
                MDAR'A          AMASK
                ARMD'B          NEWS
                MDIC'A'H        CM40
                S5AR'F'H
                MDAR'A          C7              [FOOT PEDALS
                MDAE'L
                MDAR            .+3             [FP1 - FS1 TEXT EDIT REQ
                ARIR'F                          [FP2 - FS8 GO EDIT
                JUMP            .+9.
                0;1;400;401;100000;100001;100400;100401
                NOOP
                MDAR'O          NEWS
                ARMD            NEWS
                ARMD            TOLDS           [SAVE FOR NEXT OLD SWITCHES
                MDXO'A          OLDS            [GET CHANGED SWITCHES
                ARMD            CNGS
```

```
        MDAR'F'N    2           [SETUP FRAME COUNTER
        ARMD        FTIME
        ARMD'0      FFLAG       [SET FRAME FLAG
        MDAR'H      $GTWD1
        JPLS        SA1         [JUMP IF COMMAND PRESENT

SB:     MDAR        DBUG
        JPLS        .+3
        MDAR'N      OKFLG
        JPAN        SE          [JUMP IF NO

        ARX0'F
        ARMD        OKFLG       [RESET FLAG
        MDAR        MODE
        MDX0'F      3           [GRAPHICS WAIT MODE
        JPLS        .+2
        JUMP        SK4         [JUMP IF YES
        MDX0'F      3'2         [TEXT WAIT MODE
        JPLS        .+2
        JUMP        SJ4         [JUMP IF YES

SE:     MDAR        MODE        [DISPATCH ON MODE
        MDAE'L
        JUMP        .+1
        ARIR'F
        JUMP        SG          [DISPLAY MODE
        JUMP        SF          [TEXT WAIT MODE
        JUMP        SF          [GRAPHICS WAIT MODE
        JUMP        SJ          [TEXT SELECT MODE
        JUMP        SK          [GRAPHICS SELECT MODE
        JUMP        SL          [TEXT EDIT MODE
```

```
        JUMP          SQ           [GRAPHICS EDIT MODE
SA1:    ARRS          3            [GET COMMAND
        MDAR'A'F      77
        MDXO'F        22           [SPECIAL COMMAND
        JPLS          .+4

        MD10'A'L
                      177777
        JUMP          $93COX       [GO DO SPECIAL COMMAND

        MDXO'F        22'20        [EDIT O.K.
        JPLS          SA3          [JUMP IF NOT

        ARMD'O        OKFLG        [SET 9.K. TO EDIT FLAG

SA2:    ARXO'F
        ARMD          $GTWD1       [RESET COMMAND WORD
        JPSR          $FINSH       [SET ACCESS BIT
        JUMP          SB

SA3:    MDAR'N        MODE         [O.K. TO PROCESS COMMAND
        MDAS'F        6
        JPAN          SB           [IGNORE IF NOT
        MDAR'H        $GTWD1       [GET COMMAND
        ARRS          3
        MDAR'A'F      77
        MDXO'F        1            [IS IT VCD COMMAND
        JPLS          N2           [NO, TEST FOR JOYSTICK
        JPSR          $TRVCD       [YES, GET DIAL SETTINGS
        ARXO'F
        ARMD          CNT          [SET UP LOOP TO OBTAIN
        MDAR'F        $TVCDA       [ALL SIX VALUES
```

```
          MDAE'L;       -1
          ARMD'L;
FADD:     O
          MDAR'I'X      FADD          [GET VALUES OF ALL
          ARRS          6             [DIALS
          NOOP
          ARMD'I        FADD
          MDAR'X        CNT
          MDXO'L;       6
          JPLS          FADD+1
          JPSR          $ROWFW        [GET READY TO QEND ACCROSS
                        -0            [TO 9300
                        $GTWD2
                        $TVCDA
                        6
          JPSR          FSWI          [TRANSMIT THE VALUES
                        7BLK
          JUMP          SB

N2:       MDXO'F        1'2           [IS IT JOYSTK COMMAND
          JPLS          N3            [NO, TEST FOR F.S.
          JPSR          $TRJSB
          ARXO'F
          ARMD          CNT           [SET UP LOOP TO
          MDAR'F        $TJSBX        [OBTAIN ALL VALUES
          MDAE'L;       -1
          ARMD'L;
FADD1:    O
          MDAR'X'I      FADD1
          ARRS          6
          NOOP
          ARMD'I        FADD1         [NOW GET ALL VALUES
```

```
          MDAR'X        CNT
          MDXØ'L;       3
          JPLS          FADD1+1       [GET READY TØ TRANSMIT
          JPSR          $RØWFW        [TØ THE 9300
                        -0
                        $GTWD2
                        $TJSBX
                        3
          JPSR          FSWI          [TRANSMIT VALUES
                        7BLK
          JUMP          SB

N3:       MDXØ'F        2'3           [TEST FØR F.S. CØMMAND
          JPLS          N10
          MDAR          NEWS          [GET NEW QWITCHES
          ARMD          TEMP3         [STØRE FØR TRANSFER
          JPSR          $RØWFW        [TØ 9300
                        -0
                        $GTWD2
                        TEMP3
                        1
          JPSR          FSWI          [TRANSMIT TØ 9300
                        7BLK
          JUMP          SB
N10:      MDAR'H        $GTWD1        [GET CØMMAND
          ARRS          3
          MDAR'F'A      73
          MDXØ'F        10            [RECIEVE TEXT BLØCK
          JPLS          SA4           [SKIP IF NØT

          MDAR          $GTWD3        [BLØCK 0
          JPLS          SA3A          [SKIP IF NØT
```

```
          MDAR'L;       ENDCORE
          ARMD          TPTR          [FREE UP BLOCK STORAGE
          MDAR'L;       TBLK
          ARMD          TBLK          [RESET TEXT BLOCKS
          ARX0'F
          ARMD          TBLKN
          MDAR          MODE          [IN TEXT WAIT OR SELECT
          MDAR!A'F      1
          JPLS          .+4           [SKIP IF NOT

          MDAR          C1
          ARMD          MODE          [SET DISPLAY MODE
          ARMD          CBLKP

SA3B:     JPSR          FSWI          [FILL SWI
                        1BLK
          JUMP          SB

SA3A:     MDAS'N        TBLKN         [BLOCK ALREADY HERE
          JPAN          SA3C          [JUMP IF YES

          MDAR          TPTR          [ENOUGH ROOM
          MDAS'F'N      26.
          MDAE'N        FREE
          JPAN          SA9           [JUMP IF NOT

          MDAR          TBLKE         [ROOM FOR ENTRY
          MDAE'N        TBLK
          MDAS'F'N      3
          JPAN          SA9           [JUMP IF NOT
```

```
            MDAR        TPTR        [ALLOCATE BLOCK
            MDAS'F'N    26.
            ARMD        TPTR
            MDAS'F'N    1
            ARMD'I'X    TBLK
            ARMD        T1
            MDAS'F      25.
            ARMD        T4
            MDAR        TBLK        [SET POINTER TO BLOCK
            ARMD        T2
            ARMD        T10
            MDAR        C26.        [SET SIZE
            ARMD'I'X    TBLK
            ARXO'X      TBLKN       [INDEX NO. TEXT BLOCKS
            ARMD'I'X    TBLK
SA3D:       JPSR        $ROWFW      [READ
                        0
                        $GTWD2
                        TBUFF
                        51.
            MDAR        $GTWD1      [GET COUNT
            MDAS'F'N    3
            ARMD'N      T3
            MDAR        TBUFF       [GET SIZE
            ARRS        7
            MDAR'A      CMASK
            MDAE'L
            MDAR        STBL-1
            ARIR'F
            ARMD'I'H    T1          [SET SIZE
            MDAR        TBUFF       [GET INTENSITY
```

```
          MDAR'A        CMASK
          MDAE'L
          MDAR'H        ITBL-1
          ARIR'F
          MDAR'I'0'H    T1            [C0MBINE
          ARMD'I'B      T1            [SAVE IN TABLE
          MDAR'H        TBUFF+1       [GET X P0S.
          MDAR'0        TBUFF+2       [C0MBINE WITH Y
          ARMD'I'X'B    T1            [SAVE IN LIST
          MDAR'F        TBUFF+2       [SETUP FETCH PTR.
          ARMD          T5

SA3E:     MDAR'X        T3            [ANY M0RE
          JPLS          SA3F          [SKIP IF YES

          MDAR'I        T1            [INSERT E0L
          MDAR'0'H      C1
          ARMD'I        T1

SA3G:     ARX0'F
          ARMD'I'X'B    T1            [ZER0 W0RD
          MDAR          T1            [END 0F ALL0CATI0N
          MDX0          T4
          JPLS          SA3G          [REPEAT IF N0T

SA3H:     MDAR'H'N      $GTWD1
          MDAR'A'F      40
          JPLS          SA3B          [G0 ACKN0WLEDGE IF N0T T0 EDIT

          JPSR          FSWI
                        1BLK          [RESP0ND
          ARX0'F
```

```
          ARMD          NFLAG
          MDAR          C2
          ARMD          MODE          [SET TEXT WAIT MODE
          ARMD'O        OKFLG         [SET O.K. TO EDIT
          MDAR          T10           [COMPUTE BLOCK NUMBER
          MDAE'L'N;     TBLK-2
          ARLS          1
          DIVI          3
          NOOP
          ARMD          CBLKN         [SAVE
          MDAR          T10           [GET BLOCK POINTER
          JUMP          SJ5           [GO SETUP LINE

SA3F:     MDAR'I'X      T5            [FETCH WORD
          ARMD'H        T6            [SAVE
          MDAR'X        T3            [ANY MORE
          JPLS          SA3J          [JUMP IF YES

          MDAR          T6            [GET PARTIAL WORD
          MDAR'O        CB15          [SET EOL
          JUMP          SA3G+1        [GO STORE AND FINISH

SA3J:     MDAR'I'X      T5            [FETCH NEXT WORD
          MDAR'O        T6
          ARMD'I'B'X    T1            [SAVE
          MDAR          T1            [END OF ALLOCATION
          MDXO          T4
          JPLS          SA3E          [JUMP IF NOT

          MDAR'I        T1            [INSERT EOL
          MDAR'O'H      C1
          ARMD'I        T1
```

```
        JUMP      SA3H      [G0 END

SA3C:   MDAR'B    $GTWD3    [GET 3*BL0CK
        MDAS      $GTWD3
        MDAE'L;   TBLK-2    [GET BL0CK P0INTER
        ARMD      T2
        ARMD      T10       [SAVE BL0CK P0INTER
        MDX0      CBLKP     [CURRENT BL0CK
        JPLS      .+4       [SKIP IF N0T

        MDAR      C1
        ARMD      M0DE      [SET T0 DISPLAY M0DE
        ARMD      CBLKP     [AND CLEAR CURRENT P0INTER

        MDAR'I    T2        [SET START ADDRESS
        ARMD      T1
        MDAS'F    25.       [SET END ADDR.
        ARMD      T4
        JUMP      SA3D      [G0 M0VE

SA4:    MDX0'F    10'11     [TRANSMIT TEXT T0 SDS
        JPLS      SA5       [SKIP IF N0T

        MDAR      $GTWD3    [GET BL0CK
        MDAS'N    TBLKN
        JPAN      .+2       [SKIP IF VALID

        JUMP      SA9       [ELSE, GIVE ERR0R

        MDAR'B    $GTWD3    [GET 3 * BL0CK N0.
        MDAS      $GTWD3
        MDAE'L;   TBLK-2    [GET PTR. T0 BL0CK
```

```
                ARMD            T3
                MDAR'I          T3              [GET STARTING ADDR
                ARMD            T1
                MDAR'L;         TBUFF           [SETUP STORE POINTER
                ARMD            T2
                MDAR'I          T1              [GET SIZE CODE
                JPSR            GSIZ
                ARMD'I'B        T2              [SAVE
                MDAR'I'H        T1              [GET INTENSITY
                JPSR            GINT
                MDAR'O'K        T2              [COMBINE
                ARMD'I          T2
SA4C:           MDAR'I'X'H      T1              [FETCH WORD
                ARRS            1
                MDAR'A          M1629
                ARMD'I'X        T2              [SAVE UPPER PORTION
                MDAR'I          T1              [GET WORD
                ARRS            1
                MDAR'A          M1629
                ARMD'I'X        T2              [SAVE LOWER PORTION
                MDAR'I'H'N      T1              [EOL
                MDAR'A'F        1
                JPLS            SA4C            [REPEAT IF NOT

                MDAR'I          T1              [ANY LOWER CHARACTERS
                JPLS            SA4D            [SKIP IF YES

                MDAR            T2              [BACK UP POINTER
                MDAS'F'N        1
                ARMD            T2

SA4D:           MDAR            T2              [COMPUTE LENGTH
```

```
              MDAE'L'N;     TBUFF-1
              ARMD          T5            [SAVE LENGTH

SA4E:         MDAR          T2
              MDXØ'F        TBUFF+50.
              JPLS          .+2           [SKIP IF NØT AT END

              JUMP          SA4F

              ARXØ'F
              ARMD'I'X      T2            [ZERØ WØRD
              JUMP          SA4E          [REPEAT

SA4F:         MDAR          $GTWD1        [GET NØ. WØRDS
              MDAR'A        M1629
              ARMD          .+5           [SAVE
              JPSR          $RØWFW        [WRITE
                            -0
                            $GTWD2
                            TBUFF
                            0
              JPSR          FSWI          [FILL SWI
                            2BLK          [WITH Ø.K.
              JUMP          SB


SA5:          MDXØ'F        11'12         [RECIEVE GRAPHICS FRØM SDS
              JPLS          SA6           [SKIP IF NØT

              MDAR          $GTWD3        [BLØCK = 0
              JPLS          SA5B          [SKIP IF NØT
```

```
        MDAR       $LD
        ARMD       FREE       [UNALLOCATE GRAPHICS BLOCK STORAGE
        MDAR'L;    GBLK       [RESET GRAPHICS BLOCKS
        ARMD       GBLK
        ARXO'F
        ARMD       GBLKN
        MDAR'N     MODE       [IN SELECT OR WAIT MODE
        MDAR'A'F   1
        JPLS       .+4        [SKIP IF NOT

        MDAR       C1
        ARMD       MODE       [SET DISPLAY MODE
        ARMD       CBLKP      [RESET CURRENT POINTER

SA5A:   JPSR       FSWI       [FILL SWI
                   3BLK       [WITH O.K.
        JUMP       SB

SA5B:   MDAS'N     GBLKN      [BLOCK ALREADY HERE
        JPAN       SA5C       [JUMP IF YES

        MDAR       $GTWD1     [GET COUNT
        MDAR'A     M1629
        ARMD       SA5D+2     [SAVE FOR I/O
        MDAR       TPTR       [ENOUGH CORE
        MDAE'N     FREE
        MDAS'N     SA5D+2
        JPAN       SA9        [JUMP IF NOT

        MDAR       GBLKE      [ENOUGH FOR BLOCK HEADER
        MDAE'N     GBLK
```

```
         MDAS'F'N     3
         JPAN         SA9          [JUMP IF NØT

         MDAR         FREE         [SETUP ALLØCATIØN
         ARMD         SA5D+1
         ARMD'I'X     GBLK
         MDAR         GBLK
         ARMD         T10          [SAVE BLØCK PØINTER
         MDAR'X       GBLK         [GET PØINTER
         ARMD         T4
         MDAR         FREE
         MDAS         SA5D+2
         ARMD         FREE
         MDAS'F'N     1
         ARMD'I       T4           [SAVE END PØINTER
         ARMD'I'X     GBLK
         ARAR'X       GBLKN        [INDEX BLØCK NUMBER
SA5E:    JPSR         $RØWFW       [READ DATA
                      0
SA5D:                 $GTWD2
                      0
                      0
         MDAR'I       SA5D+1       [SETUP INTENSITY
         ARLS         3
         ARMD         T3
         ARXØ'F
         MDAS         T3
         ARRS         1
         MDAR'A       AMASK
         ARMD         T3
         MDAR'I       SA5D+1
         ARLS         3
```

```
         MDAR'A'H     C1          [GET DASH BIT
         MDAR'O       T3          [COMBINE
         ARMD         T3          [STORE IN T3
         MDAR'I'K     SA5D+1      [SET UP SCALE
         MDAR'H'A     SCMASK      [MASK
         MDAR'O       T3          [COMBINE
         ARMD'I       SA5D+1      [SAVE
         MDAR'I'K'X   SA5D+1      [GET DX
         MDAR'H'A     SCMASK      [MASK
         ARMD         T5
         MDAR'I       SA5D+1      [SAVE
         ARLS         3
         MDAR'A'L;    77777       [GET DY
         MDAR'O       T5          [COMBINE
         ARMD'I       SA5D+1      [SAVE
SA5F:    MDAR'I'K'X   SA5D+1      [FETCH X PART
         MDAR'A       M0010       [MASK
         ARMD         T5
         MDAR'I       SA5D+1      [FETCH Y PART
         ARLS         3
         MDAR'A'H     M0010       [MASK
         MDAR'O       T5          [COMBINE
         ARMD         T5          [SAVE
         MDAR'I       SA5D+1      [GET MOVE/DRAW
         MDAR'A       C1
         MDAR'O       T5          [COMBINE
         ARMD'I       SA5D+1      [RESTORE IN BLOCK
         MDAR         SA5D+1      [END OF BLOCK
         MDXO'I       T4
         JPLS         SA5F        [REPEAT IF NOT

SA5G:    MDAR'I       SA5D+1      [INSERT EOL
```

```
                MDAR'Ø'H        C1
                ARMD'I          SA5D+1
                MDAR'H'N        $GTWD1
                MDAR'A'F        40              [LØAD AND EDIT
                JPLS            SA5A            [JUMP IF NØT

                JPSR            FSWI            [RESPØND
                                3BLK
                ARXØ'F
                ARMD            NFLAG           [RESET NEW BLØCK FLAG
                MDAR            C3
                ARMD            MØDE            [SET GRAPHICS WAIT MØDE
                ARMD'Ø          ØKFLG           [SET Ø.K. TØ EDIT
                MDAR            T10             [CØMPUTE BLØCK NUMBER
                MDAE'L'N;       GBLK-2
                ARLS            1
                DIVI            3
                NØØP
                ARMD            CBLKN
                MDAR            T10             [FETCH BLØCK PØINTER
                JUMP            SK5             [GØ SETUP CURRENT BLØCK

SA5C:           MDAR'B          $GTWD3          [GET 3 * BLØCK
                MDAS            $GTWD3
                MDAE'L;         GBLK-2          [GET PØINTER
                ARMD            T1
                ARMD            T10
                MDAS'F          1
                ARMD            T4
                MDAS'F          1
                ARMD            T2
                MDAR'I          T1              [GET STARTING ADDR
```

```
            ARMD         SA5D+1       [SAVE
            MDAS         $GTWD1       [ADD LENGTH
            MDAE'N'I     T2           [CUBTRACT ALLØCATIØN END
            MDAS'F'N     1
            JPAN         .+2          [JUMP IF FITS

            JUMP         SA9          [JUMP IF NØT ENØUGH RØØM

            MDAR         $GTWD1       [GET LENGTH
            MDAR'A       M1629
            ARMD         SA5D+2       [SAVE
            MDAE'I       T1           [ADD ØRIGIN
            MDAS'F'N     1            [CREATE END ADDR.
            ARMD'I       T4           [SAVE
            JUMP         SA5E         [GØ READ

SA6:        MDXØ'F       12'13        [TRANSMIT GRAPHICS TØ SDS
            JPLS         SA2          [IGNØRE IF NØT

            MDAR         $GTWD3       [GET BLØCK NØ.
            MDAS'N       GBLKN
            JPAN         .+2          [JUMP IF Ø.K.

            JUMP         SA9          [JUMP IF INVALID

            MDAR'B       $GTWD3       [GET 3 * BLØCK NØ.
            MDAS         $GTWD3
            MDAE'L;      GBLK-2       [GET PØINTER
            ARMD         T1
            MDAS'F       2
            ARMD         T2
            MDAR'I       T1           [GET STARTING ADDR. - 1
```

```
            ARMD          T3
            MDAR'I        T3              [FETCH FIRST WORD
            MDAR'A'H      C1              [EXTRACT DASH BIT
            ARRS          3
            ARMD          T5              [SAVE
            MDAR'I        T3              [GET INTEN.
            ARRS          2
            MDAR'A'L,     7777
            MDAR'O        T5
            ARMD          T5              [SAVE
            MDAR'I        T3              [GET SCALE
            MDAR'H'A      SCMASK          [MASK
            ARRS          6
            MDAR'O        T5
            ARMD          T5              [SAVE
            JPSR          $ROWFW          [WRITE FIRST WORD
                          -0
                          $GTWD2
                          T5
                          1
            MDAR          $GTWD1          [GET NO WORDS
            MDAR'A        M1629
            MDAS'F'N      1
            ARMD          T5
            MDAR'X        $GTWD2          [GET BLOCK ADDR
            MDAR          C50.            [SET 50. WORD LENGTH
            ARMD          SA6B+2

SA6A:       MDAR'F'N      49.             [SETUP WORD COUNT
            ARMD          T6
            MDAR'F        TBUFF-1
            ARMD          T7              [SET STORE POINTER
```

```
SA6C:     MDAR'I'H'X    T3         [FETCH LEFT PØRTIØN
          ARLS          3
          MDAR'A'K      M1828
          ARMD'I'X'K    T7         [SAVE
          MDAR'I        T3         [FETCH RIGHT PART
          ARRS          3
          MDAR'A        M1828
          MDAR'Ø'I      T7         [CØMBINE
          ARMD'I        T7
          MDAR'I        T3         [GET MØVE/DRAW
          MDAR'A        C1
          MDAR'Ø'I      T7         [CØMBINE
          ARMD'I        T7         [SAVE
          MDAR'X        T6         [END ØF SECTØR
          JPLS          SA6C       [REPEAT IF NØT

          MDAR'N        T5         [CØUNT LESS THAN 50
          MDAS'F        50.
          JPAN          .+3        [SKIP IF NØT

          MDAR          T5
          ARMD          SA6B+2     [SET LØWER CØUNT

          JPSR          $RØWFW     [WRITE SECTØR
                        -0
SA6B:                   $GTWD2
                        TBUFF
                        50.
          MDAR          $GTWD2
          MDAS'F        50.
          ARMD          $GTWD2
```

```
                MDAR        T5              [REDUCE REMAINING COUNT
                MDAS'F'N    50.
                ARMD        T5
                JPAN        .+2             [SKIP IF DONE

                JUMP        SA6A            [ELSE, REPEAT

                JPSR        FSWI            [FILL SWI
                            4BLK            [WITH DONE RESPONCE
                JUMP        SB



SA9:            JPSR        FSWI            [FILL SWI WITH ERROR CODE
                            5BLK
                JUMP        SB              [RETURN
[ DISPLAY MODE ENTRY

SG:             IFCHANGE (((1,SD),(2,SC)))
                JUMP        SF


SD:             MDAR        C4              [SET TEXT SELECT MODE
                ARMD        MODE
                ARMD        NFLAG           [RESET NEW FLAG
                MDAR        TBLK            [ANY BLOCKS
                MDXO'F      TBLK
                JPLS        .+2             [SKIP IF YES

                JUMP        SJ3             [ELSE, SETUP NEW BLOCK
```

```
         MDAR      C1        [SET BLOCK 1
         ARMD      CBLKN
         MDAR'L              [SETUP BLOCK 1 POINTER
                   TBLK+1
         JUMP      SJ5

SC:      MDAR      C5        [SET GRAPHICS SELECT MODE
         ARMD      MODE
         ARMD      NFLAG     [RESET NEW BLOCK FLAG
         MDAR      GBLK      [BLOCK 1 PRESENT
         MDXO'F    GBLK
         JPLS      .+2       [SKIP IF YES

         JUMP      SK3       [ELSE, SET NEW BLOCK

         MDAR      C1        [SET BLOCK 1
         ARMD      CBLKN
         MDAR'L              [SET POINTER
                   GBLK+1
         JUMP      SK5       [GO PROCESS

SZ:      MDAR      C1        [SET DISPLAY MODE
         ARMD      MODE
         ARMD      CBLKP     [RESET CURRENT POINTER
         JUMP      SF        [GO DISPLAY

SG1:     MDAR      C2        [TEXT EDIT REQUEST
```

```
SG3:        ARMD        MODE        [SET MODE
            JPSR        WSWI        [WRITE SWI
                        6BLK        [FOR EDIT REQUEST
            JUMP        SF          [GO DISPLAY


SG2:        MDAR        C3          [GRAPHICS EDIT REQUEST
            JUMP        SG3         [GO PROCESS
[ DISPLAY ROUTINE ENTRANCE

SF:         MDAR'F      GBLK        [SET POINTER TO GRAPHICS BLOCKS

SF1:        ARMD        BLKP
            MDX0        GBLK        [END
            JPLS        SF2         [JUMP IF NOT

            MDAR'F      TBLK        [SET START OF TEXT BLOCKS

SF4:        ARMD        BLKP
            MDX0        TBLK        [END
            JPLS        SF5         [JUMP IF NOT

            JPSR        PTRAC       [TRACK PEN
            MDAR'L                  [GET ADDR. OF MODE MESSAGE
            MDAR        TXLIN-1
            MDAS        MODE
            ARIR'F
            JPSR        DTEXT       [DISPLAY MODE MESSAGE
            MDAR        ETIME       [ERROR MESSAGE PRESENT
            JPLS        SF4A        [JUMP IF YES

SF6:        MDAR        FFLAG       [FRAME FINISHED
```

```
                JPAN          •-1          [REPEAT IF NØT

                JUMP          SA           [GØ START NEW FRAME

SF4A:           MDAR'X        ETIME        [INDEX TIMER
                MDAR'A'F      10           [TIME TØ DISPLAY
                JPLS          SF6

                MDAR'F        ERMSG        [DISPLAY MESSAGE
                JPSR          DTEXT
                JUMP          SF6          [GØ END

SF2:            MD10'L'Ø      .            [TURN ØN VECTØR GENERATØR
                60740√H
                MDAR'I'X      BLKP         [FETCH STARTING ADDR
                MDAR'Ø'L
                MD05
                ARMD          EØVPV        [SET VECTØR PIVØT
                MDAR'H        AMASK        [LØAD MASK
                MDAR'I'A      EØVPV        [SET SCALE
                AR11
                MD06'I        EØVPV        [SET INTENSITY
                MDAR'I'H'N    EØVPV        [FETCH DASH BIT
                MDAR'A'F      1
                JPLS          •+3          [SKIP IF NØT ØN

                MD10'Ø'L;     2000√H       [TURN ØN DASH
                MD07'I'X      EØVPV        [SET DX, DY
                MDAR          BLKP         [FETCH BLØCK PØINTER
                MDXØ          CBLKP        [CURRENT BLØCK
                JPLS          SF3          [SKIP IF NØT
```

```
                MD06'F          17777           [SET HIGH INTENSITY
                MDAR'F          PENS            [ENABLE LIGHT PEN
                ARMD            LPNPV
                MD10'O'L
                36/H

SF3:            ARXO'F
                ARMD            DSDON           [RESET DISPLAY DONE FLAG
                MDAR'F          EOLS            [SET EOL INTERRUPT
                ARMD            EOLPV
                MDIR'X          EOVPV           [STRT DISPLAY

                MDAR'N          DSDON           [DONE
                JPAN            .-1

                MDAR'F          NULX            [NULL LIGHT PEN
                ARMD            LPNPV
                MD10'A'L                        [TURN OFF AVG
                01740/H
                MDAR            BLKP            [CURRENT BLOCK
                MDXO            CBLKP
                JPLS            .+2

                JPSR            PTRAC           [TRACK PEN IF YES

                MDAR            BLKP            [UP POINTER
                MDAS'F          2
                JUMP            SF1             [GO TRY AGAIN

SF5:            MDAR'X          BLKP            [INDEX POINTER
                MDXO            CBLKP           [CURRENT BLOCK
                JPLS            SF7             [JUMP IF NOT
```

```
                MDAR'F          CRLIN-1         [DISPLAY CURRENT LINE
                JPSR            DTEXT
                JUMP            SF8

SF7:            MDAR'I          BLKP            [DISPLAY NØRMAL LINE
                JPSR            DTEXT

SF8:            MDAR            BLKP            [INCREMENT PØINTER
                MDAS'F          2
                JUMP            SF4             [GØ TRY AGAIN
[ ENTRY FØR TEXT SELECT MØDE

SJ:             IFCHANGE ((( 5,SJ1),(6,SJ2),(7,SJ3),(8,SG1),(4,SZ)))
                JUMP            SF


SJ4:            MDAR            C6              [SET TEXT EDIT MØDE
                ARMD            MØDE
                MDAR            CBLKN           [GET BLØCK NØ. ASCII
                JPSR            GASCI
                ARMD            TXL6A
                MDAR            CRLIN           [GET SIZE ASCII
                JPSR            GSIZ
                JPSR            GASCI
                ARMD            TXL6B           [SAVE
                MDAR            CRLIN+1         [GET X PØS ASCII
                ARRS            1
                MDAR'A          CMASK
                MDAS'F'N        15.
                JPSR            GASCI
                ARMD            TXL6C
```

```
          MDAR        CRLIN+2     [GET Y P0S ASCII
          ARRS        1
          MDAR'A      CMASK
          MDAS'F'N    22.
          ARRS        1
          JPSR        GASCI
          ARMD        TXL6D
          MDAR'B      C137        [INSERT CURSAR
          ARMD        CRLIN+3
          MDAR'F      CRLIN+3     [SET P0INTER
          ARMD        CP0S
          MDAR'F'N    7           [SETUP CURSAR C0UNTER
          ARMD        CURCT
          MDAR'F'N    4
          ARMD        REPCT       [SET REPEAT C0UNTER
          MDAR'N      NFLAG       [NEW BL0CK
          JPAN        SJ4A        [JUMP IF N0T

          MDAR        TBLK        [UP ALL0CATI0N P0INTER
          MDAS'F      3
          ARMD        TBLK
          MDAR        TPTR
          MDAS'F'N    26.
          ARMD        TPTR
          ARAR'X      TBLKN

SJ4A:     MDAR'L                  [SETUP F0REGR0UND/BACKGR0UND
          JUMP        SF
          ARMD        $WT1
          ARX0'F
          ARMD        CFLAG       [RESET CHARACTER FLAG
SJ4B:     JPSR        $ICHTY      [INPUT CHARACTER
```

```
                         2
          MDBR           $TTYC          [FETCH CHARACTER
          MDBR'A         CMASK
          BRMD           CHAR           [SAVE CHARACTER
          ARMD'0         CFLAG          [SET CHARACTER FLAG
          JUMP           SJ4B           [GO GET ANOTHER


SJ1:      MDAR           TBLKN          [ANY MORE BLOCKS
          MDX0           CBLKN
          JPLS           .+2            [SKIP IF YES

          JUMP           SF

          MDAR           NFLAG          [NEW BLOCK
          JPAN           SF             [SKIP IF YES

          ARAR'X         CBLKN          [INDEX BLOCK NUMBER
          MDAR           CBLKP          [UP POINTER
          MDAS'F         3

SJ5:      ARMD           CBLKP
          MDAR'I         CBLKP          [FETCH ADDR. OF BLOCK
          ARMD           T1
          MDAR'F         CRLIN+3        [SETUP LINE POINTER
          ARMD           T2
          MDAR'I         T1             [FETCH FIRST WORD
          ARMD           CRLIN          [SAVE INT. AND SIZE
          MDAR'I'X'H     T1             [FETCH X POS
          MDAR'A         M1528          [MASK
          ARMD           CRLIN+1
          MDAR'I         T1             [SETUP Y POS.
```

```
            MDAR'A        M1528
            ARMD          CRLIN+2
SJ9:        MDAR'I'H      T1              [CHECK FØR EØL
            MDAR'A'F      1
            JPLS          SJ9A

            MDAR'I'X'K    T1              [GET FIRST CHARACTER
            ARLS          2
            MDAR'A        M2228
            ARMD'I'X      T2              [SAVE
            MDAR'I'H      T1              [GET SECØND CHAR.
            MDAR'A        M2228
            ARMD'I'X      T2              [SAVE
            MDAR'I'K      T1              [GET THIRD CHAR.
            ARLS          2
            MDAR'A'H      M2228
            ARMD'I'X'H    T2              [SAVE
            MDAR'I        T1              [GET FØURTH CHAR.
            MDAR'A        M2228
            ARMD'I'X      T2              [SAVE
            JUMP          SJ9             [GØ REPEAT

SJ9B:       ARXØ'F                        [STØRE NULLS
            ARMD'I'X      T2

SJ9A:       MDAR          T2              [END ØF LIST
            MDXØ'F        CRLIN+100.
            JPLS          SJ9B            [REPAAT IF NØT

SJ5A:       MDAR'H        C1              [STØRE EØL
            ARMD          CRLIN+100.
            ARXØ'F
```

```
            ARMD        CRLIN+3     [INSERT CURSAR
            MDAR        CBLKN       [GET ASCII F0R BL0CK NUMBER
            JPSR        GASCI
            ARMD        TXL4A
            JUMP        SF


SJ2:        MDAR        CBLKP       [BL0CK 1
            MDX0'F      TBLK+1
            JPLS        .+2         [SKIP IF N0T
            JUMP        SF

            ARMD'H      NFLAG       [RESET NEW BL0CK FLAG
            MDAR        CBLKN       [DECREMENT CURRENT BL0CK
            MDAS'F'N    1
            ARMD        CBLKN
            MDAR        CBLKP       [DECREMENT BL0CK P0INTER
            MDAS'F'N    3
            JUMP        SJ5         [G0 PR0CESS

SJ3:        MDAR        TPTR        [R00M
            MDAE'N      FREE
            MDAS'F'N    27.
            JPAN        SJ6         [JUMP IF N0T

            MDAR        TBLKE       [R00M F0R ENTRY
            MDAE'N      TBLK
            MDAS'F'N    3
            JPAN        SJ6         [JUMP IF N0T

            MDAR        TBLK        [SETUP ALL0CATI0N
            MDAS'F      1
```

```
ARMD            T1
ARMD            CBLKP
MDAR            TPTR
MDAS'F'N        27.             [SETUP ADDRESS
ARMD'I          T1
ARMD'O          NFLAG           [SET NEW BLOCK FLAG
MDAR            TBLKN
MDAS'F          1               [GET NEXT BLOCK NO.
ARMD            CBLKN           [SAVE
MDAR            C26.
ARMD'I'X        T1              [SET LENGTH
ARXO'F
ARMD'I'X        T1
MDAR'L                          [SETUP CLEAR COMMAND
ARMD            CRLIN
ARMD            .+2             [SAVE

ARXO'F
NOOP                            [RESET WORD
MDAR'X          .-1             [INDEX
MDXO'F          CRLIN+100.
JPLS            .-4             [REPEAT IF NOT END

MDAR'L                          [SETUP X POS
P (0,0,11,20)
ARMD            CRLIN+1
MDAR'L                          [SETUP Y POS
P (0,0,13,24.)
ARMD            CRLIN+2
MDAR'L                          [SETUP SIZE
P (0,0,0,22)
ARMD            CRLIN
```

```
               JUMP           SJ5A          [GO FINISH

SJ6:SK6: MDAR'F'N             200.          [SET ERROR MESSAGE TIMER
               ARMD'L
ETIME:                        0
               JUMP           SF            [GO DISPLAY
[ GRAPHICS SELECT MODE PROCESSOR

SK:            IFCHANGE (((5,SK1),(6,SK2),(7,SK3),(8,SG2),(4,SZ)))
               JUMP           SF            [DISPLAY IF NO CHANGE

SK4:           MDAR           C7            [SET GRAPHICS EDIT MODE
               ARMD           MODE
               MDAR           COFF          [SET CURSAR  OFF
               ARMD           TXL7B
               MDAR           CBLKN         [GET BLOCK NO. TEXT
               JPSR           GASCI
               ARMD           TXL7A
               ARXO'F
               ARMD           TXL7C         [RESET FULL
               ARMD           CURFG         [RESET CURSAR ON FLAG
               ARMD           SVFLG         [RESET SAVED VECTOR FLAG
               ARMD           TEPFG         [AND TRACK ENDPOINT MODE FLAG
               MDAR           CBLKP         [SET POINTERS TO ENTRY
               MDAR'A         AMASK
               MDAS'F         1
               ARMD           CREND
               MDAS'F         1
               ARMD           ABEND
               MDAR'I         CREND         [FULL
               MDXO'I         ABEND
```

```
                JPLS        .+3         [SKIP IF NØT

                MDAR        CFULL       [SET  FULL  IN MESSAGE
                ARMD        TXL7C

                MDAR'N      NFLAG       [NEW BLØCK
                JPAN        SF          [JUMP IF NØT

                MDAR        GBLK        [FREEZE ALLØCATIØN
                MDAS'F      3
                ARMD        GBLK
                MDAR'I      CBLKP       [SET CURRENT END
                MDAS'F      2
                ARMD'I      CREND
                MDAS'F'N    2
                ARMD        CPTR
                MDAR        TEMP2
                ARMD'I      CPTR
                MDAR        TEMP1
                ARMD'I'X    CPTR        [DX,DY = 0
                MDAR'H      C1          [SET EØL IN LIST
                ARMD'I'X    CPTR
                MDAR        TPTR        [SET MAX END
                ARMD'I      ABEND
                ARAR'X      GBLKN       [INDEX BLØCK NØ.
                JUMP        SF          [GØ DISPLAY

SK1:            MDAR        GBLKN       [ANY MØRE BLØCKS
                MDXØ        CBLKN
                JPLS        .+2         [SKIP IF YES
                JUMP        SF          [ELSE, DISPLAY
```

```
            MDAR        NFLAG       [NEW BLOCK
            JPAN        SF          [DISPLAY IF YES

            ARAR'X      CBLKN       [INDEX CURRENT BLOCK NO.
            MDAR        CBLKP       [INCREMENT POINTER
            MDAS'F      3

SK5:        ARMD        CBLKP       [SAVE POINTER
            MDAR        CBLKN       [GET ASCII FOR BLOCK NO.
            JPSR        GASCI
            ARMD        TXL5A       [SAVE IN MESSAGE
            JUMP        SF          [GO DISPLAY

SK2:        MDAR        CBLKN       [BLOCK 1 CURRENT
            MDXO'F      1
            JPLS        .+2         [SKIP IF NOT

            JUMP        SF          [ELSE, GO DISPLAY

            ARMD'H      NFLAG       [RESET NEW BLOCK FLAG
            MDAR        CBLKN       [DECREMENT BLOCK NO.
            MDAS'F'N    1
            ARMD        CBLKN
            MDAR        CBLKP       [DECREMENT BLOCK POINTER
            MDAS'F'N    3
            JUMP        SK5         [GO SET NEW BLOCK

SK3:        MDAR        TPTR        [FETCH CORE LIMIT
            MDAE'N      FREE
            MDAS'F'N    10.         [SEE IF 10 WORDS AVAILABLE
            JPAN        SK6         [JUMP IF NOT
```

```
          MDAR          GBLKE          [ROOM FOR ENTRY
          MDAE'N        GBLK
          MDAS'F'N      3
          JPAN          SK6            [JUMP IF NOT

          MDAR          GBLK           [ALLOCATE BLOCK (TEMPORARY)
          MDAS'F        1
          ARMD          T1
          ARMD          CBLKP
          MDAR          FREE           [SET START OF BLOCK
          ARMD'I        T1
          ARMD'O        NFLAG          [SET NEW BLOCK FLAG
          MDAR          GBLKN          [SET CURRENT BLOCK NO
          MDAS'F        1
          ARMD          CBLKN
          JUMP          SK5+1          [GO PROCESS
[TEXT EDIT MODE PROCESSOR

SL:       MDAR'N        CFLAG          [CHARACTER PRESENT
          JPAN          SM             [JUMP IF NOT

          ARMD          CFLAG          [RESET FLAG
          MDAR          CHAR           [CHARACTER = RUBOUT
          MDXO'F        177
          JPLS          SL1            [JUMP IF NOT

          MDAR          CPOS           [CURSAR AT BEG. OF LINE
          MDXO'F        CRLIN+3
          JPLS          .+2            [SKIP IF NOT

          JUMP          SM             [SKIP MOVE IF YES
```

```
        JPSR        PSHDN          [M0VE BUFFER D0WN
        MDAR        CP0S           [BACK UP CURSAR
        MDAS'F'N    1
        ARMD        CP0S
        MDAR        C40            [INSERT CURSAR CHARACTER
        ARMD'I'B    CP0S
        JUMP        SM             [G0 CHECK SWITCHES

SL1:    MDX0'F      177'5          [CHAR = CNTL  E
        JPLS        SL2            [JUMP IF N0T

        JPSR        PSHDN          [M0VE CHARACTERS D0WN
        MDAR        C40            [RE-INSERT CURSAR
        ARMD'I'B    CP0S
        JUMP        SM             [G0 GET SWITCHES

SL2:    MDX0'F      5'6            [CHAR = CNTL  F
        JPLS        SL3            [JUMP IF N0T

        MDAR        CP0S           [CURSAR AT END 0F LINE
        MDAS'F      1
        ARMD        T1
        MDX0'F      CRLIN+100.
        LSAR'I      T1             [0R CHARACTER AFTER = NULL
        JPLS        .+2            [SKIP IF 0.K.

        JUMP        SM             [ELSE, READ SWITCHES

        MDAR'I      T1             [SWAP CHARACTERS
        ARMD'I      CP0S
        MDAR        C40            [INSERT CURSAR
        ARMD'I'X'B  CP0S
```

```
                JUMP            SM              [GO CHECK SWITCHES

SL3:            MDXO'F          6'22            [CHAR = CNTL  R
                JPLS            SL35            [SKIP IF NOT

                MDAR            CPOS            [CURSAR AT BEG. OF LINE
                MDAS'F'N        1
                ARMD            T1
                MDXO'F          CRLIN+2
                JPLS            .+2             [SKIP IF NOT

                JUMP            SM

                MDAR'I          T1              [SWAP CHARACTERS
                ARMD'I          CPOS
                MDAR            T1              [MOVE CURSAR FORWARD
                ARMD            CPOS
                MDAR'B          C40
                ARMD'I          CPOS            [INSERT CURSAR
                JUMP            SM

SL35:           MDXO'F          22'15           [CHAR= CAR. RET.
                JPLS            SL4             [SKIP IF NOT
                JUMP            SU              [ACT LIKE AN END EDIT

SL4:            MDAR            CHAR            [CHECK VALID DISPLAY CHAR.
                MDAR'A'F        140
                LSXO            .-1
                JPLS            .+2             [SKIP IF O.K.

                JUMP            SM              [ELSE, IGNORE
```

```
               MDAR           CRLIN+99.      [BUFFER FULL
               JPLS           SM             [SKIP IF YES

               MDAR'F         CRLIN+99.      [SETUP MØVE PØINTERS
               ARMD           T1
               MDAR'F         CRLIN+98.
               ARMD           T2

SL5:           MDAR'I         T2
               ARMD'I         T1             [MØVE CHARACTER UP
               MDAR           T2
               MDAS'F'N       1              [DECREMENT PØINTERS
               ARMD           T2
               MDAR           T1
               MDAS'F'N       1
               ARMD           T1
               MDXØ           CPØS           [END ØF MØVE
               JPLS           SL5            [JUMP IF NØ

               MDAR           CHAR           [INSERT CHARACTER
               ARMD'I'B       CPØS
               MDAR           C40            [INSERT CURSAR
               ARMD'I'X'B     CPØS
               MDAR           CRLIN+99.      [BUFFER FULL
               JPLS           .+2            [SKIP IF YES

               JUMP           SM

               MDAR           CFULL
               ARMD           TXL6E          [INSERT  FULL  IN MESSAGE

SM:            IFSWITCH (((12,SM1)))
```

```
                JUMP            SN                  [JUMP IF 12. NOT ON

SM1:            MDAR'X          REPCT               [INDEX REPEAT COUNTER
                JPLS            SN                  [JUMP IF NOT END OF COUNT

                MDAR'F'N        4                   [SET COUNTER
                ARMD            REPCT
                ARXO'F
                ARMD            TOLDS               [RESET SWITCH SAVE

SN:
                IFCHANGE (((4,SU),(9,SN1),(10,SN2),(13,SN3)))
                IFCHANGE (((14,SN4),(15,SN5),(16,SN6)))
                JUMP            SP                  [JUMP IF NO CHANGE

SN1:            MDAR            CRLIN               [CURRENT SIZE = 3
                JPSR            GSIZ
                MDXO'F          3
                JPLS            .+2                 [SKIP IF NOT

                JUMP            SP

                MDXO'F          3                   [RESTORE
                MDAE'L)         STBL

SN1A:           ARMD            T1                  [SET POINTER TO SIZE
                MDAR            CRLIN               [MASK OUT OLD
                MDAR'A'H        AMASK
                MDAR'O'I'B      T1                  [INSERT NEW
                ARMD            CRLIN               [SET BACK IN LIST
                JPSR            GSIZ                [GET NEW NUMBER
                JPSR            GASCI               [GET ASCII STRING
```

```
          ARMD          TXL6B          [SET IN MESSAGE
          JUMP          SP

SN2:      MDAR          CRLIN          [GET SIZE
          JPSR          GSIZ
          MDX0'F        1              [SIZE 1
          JPLS          .+2            [JUMP IF NOT

          JUMP          SP

          MDX0'F        1              [RESTORE
          MDAE'L;       STBL-2
          JUMP          SN1A           [GO SET NEW SIZE

SN3:      MDAR'F        SXT            [SET POINTER FOR X INCREASE

SN3A:     ARMD          T1
          MDAR          CRLIN+1        [GET X POS
          MDAR'A'B      CMASK
          MDX0'I        T1             [AT END
          JPLS          .+2            [SKIP IF NOT

          JUMP          SP

          MDAR          CRLIN+1        [MOVE POINTER
          MDAS'I'H      T1
          ARMD          CRLIN+1
          ARRS          1
          MDAR'A        CMASK          [MASK POSITION
          MDAS'F'N      15.            [OFFSET
          JPSR          GASCI          [GET ASCII FOR MESSAGE
          ARMD          TXL6C
```

```
          JUMP        SP

SN4:      MDAR'F      SXT+1         [SET FØR NEGATIVE X MØVE
          JUMP        SN3A

SXT:      2√H         (96.+15.√B)
          77775√H     (16.√B)

SN5:      MDAR'F      SYT           [SET FØR PØSITIVE Y MØVE

SN5A:     ARMD        T1
          MDAR        CRLIN+2       [GET PØSITIØN
          MDAR'A'B    CMASK
          MDXØ'I      T1            [AT END
          JPLS        .+2           [SKIP IF NØT

          JUMP        SP

          MDAR        CRLIN+2       [ADJUST
          MDAS'I'H    T1
          ARMD        CRLIN+2
          ARRS        1
          MDAR'A      CMASK
          MDAS'F'N    22.
          ARRS        1
          JPSR        GASCI         [GET ASCII FØR MESSAGE
          ARMD        TXL6D
          JUMP        SP

SN6:      MDAR'F      SYT+1         [SET FØR NEGATIVE Y MØVE
          JUMP        SN5A
```

```
SYT:            4√H             (22•+80•√B)
                77773√H         (24•√B)

SP:             MDAR'X          CURCT           [INDEX CURSAR BLINK CØUNTER
                JPLS            SF              [GØ DISPLAY IF NØT ZERØ

                MDAR'F'N        7
                ARMD            CURCT           [RESET CØUNTER
                MDAR'I          CPØS            [CØMPLEMENT CHARACTER
                MDXØ'L;         (40'137√B)
                ARMD'I          CPØS
                JUMP            SF              [GØ DISPLAY


SU:             MDAR'I          CBLKP           [SETUP PØINTER TØ BLØCK
                ARMD            T1
                MDAR'L;         CRLIN+3         [SET PØINTER TØ LINE
                ARMD            T2
                MDAR'L                          [SETUP SWI WØRD 1
                00210√H         2
                ARMD            SU9A
                MDAR            CRLIN           [SAVE INT• AND SIZE
                ARMD'I          T1
                MDAR'H          CRLIN+1         [SAVE X AND Y PØS
                MDAR'Ø          CRLIN+2
                ARMD'I'X        T1

SU1:            MDAR            T2              [END ØF LIST
                MDAE'N'L;       CRLIN+98•
                JPAN            SU2             [JUMP IF NØT

SU3:            MDAR'I          T1              [INSERT EØL
```

```
              MDAR'O'H      C1
              ARMD'I        T1
              MDAR          CRLIN          [GET SIZE
              JPSR          GSIZ
              ARLS          10.            [SHIFT
              NOOP
              MDAR'O        CBLKN          [OR IN BLOCK NUMBER
              ARMD          SU9A+1         [SAVE FOR TRANSMIT
              MDAR          CRLIN+1        [GET X POSITION
              MDAR'A'B      CMASK
              ARMD'K        SU9A+2
              MDAR          CRLIN+2        [GET Y POSITION
              MDAR'A'B      CMASK
              MDAR'O'K      SU9A+2         [COMBINE
              ARMD          SU9A+2         [SAVE FOR TRANSMIT
              JPSR          WSWI           [WRITE SWI BUFFER
                            SU9A
              MDAR          C1             [SET DISPLAY MODE
              ARMD          MODE
              ARMD          CBLKP          [RESET CURRENT POINTER
              JUMP          SF             [GO DISPLAY

SU2:          JPSR          SUS            [GET WORD
              JPLS          .+2            [SKIP IF O.K.

              JUMP          SU3            [ELSE, GO END

              ARAR'X        SU9A           [INDEX WORD COUNTER
              ARMD'B        T3             [SAVE WORD
              ARAR'X        T2
              JPSR          SUS            [GET NEXT CHAR
              MDAR'O'K      T3
```

```
                ARAR'B'X        T2              [INDEX AND SHIFT
                ARMD'B          T3
                JPSR            SUS             [GET NEXT CHARACTER
                JPLS            .+2             [SKIP IF N0T ZER0

                JUMP            .+2

                ARAR'X          SU9A            [INDEX C0UNT IF N0N-ZER0

                MDAR'0'K        T3              [C0MBINE
                ARMD'B          T3
                ARAR'X          T2              [INDEX
                JPSR            SUS             [GET NEXT CHAR.
                MDAR'0'K        T3              [C0MBINE
                ARMD'I'X        T1              [SAVE IN BUFFER
                ARAR'X          T2              [INDEX P0INTER F0R NEXT
                JUMP            SU1             [G0 TRY AGAIN


SUS:            JUMP            .               [GET W0RD SUBR.
                MDAR            T2              [AT CURSAR P0SITI0N
                MDX0            CP0S
                JPLS            .+2             [SKIP IF N0T

                ARAR'X          T2              [INDEX P0INTER IF YES

                MDAR'I          T2              [FETCH CHARACTER
                MDAR'A'B        CMASK           [MASK
                MDIR            SUS             [RETURN
SU9A:                           0
```

```
                                0
                                0


[ GRAPHICS EDIT MØDE PRØCESSØR

SQ:             JPSR            PTRAC           [TRACK CURSAR IF ØN
                IFCHANGE (((4,SV),(9,SQ1),(10,SQ2),(11,SQ3)))
                IFCHANGE (((13,SQ4),(14,SQ5),(15,SQ6),(12,SQ9)))
                IFSWITCH (((1,LTRAN),(2,RTRAN),(5,UTRAN),(6,DTRAN)))
                IFSWITCH (((3,BIG),(7,SMALL)))
                IFSWITCH (((8,RØTAT)))
                JUMP            SR

SQ1:            MDAR'I          CREND           [SET LAST CØØRD AS TRACKING CØØRD.
                ARMD            T1
                MDAR'I          T1
                ARMD            TRCRD
                ARMD'Ø          CURFG           [SET CURSAR ØN FLAG
                MDAR            CØN
                ARMD            TXL7B
                JUMP            SR

SQ2:            ARXØ'F
                ARMD            CURFG           [RESET CURSAR
                MDAR            CØFF
                ARMD            TXL7B
                JUMP            SR

SQ3:            MDAR'N          SVFLG           [VECTØR SELECTED
                JPAN            SR              [JUMP IF NØT
```

```
            MDAR        SVPTR       [SAVE SELECTED VECTOR ADDR.
            ARMD        TEPTR
            JPSR        RSV         [RESET SELECTED VECTOR
            ARMD'0      TEPFG       [SET TRACK ENDPOINT MODE
            ARMD'0      CURFG       [AND CURSAR ON
            MDAR        CON
            ARMD        TXL7B
            MDAR'I      TEPTR       [FETCH VECTOR TO TRACK
            ARMD        TRCRD       [SET FOR CURSAR
            MDAR'A      C1H1
            ARMD        TEPMB       [SAVE MODE BITS
            JUMP        SR

SQ6:        MDAR        TRCRD       [SET SKETCH COORDINATES
            ARMD        SKCRD
            JUMP        SR

SQ7:        NOOP
            JUMP        SR

SQ4:        ARX0'F                  [SET DRAW BIT

SQ4D:       ARMD        T3
            MDAR'N      CURFG       [CURSAR ON
            JPAN        SS          [JUMP IF NOT

SQ4C:       MDAR'I      CREND       [FULL
            MDX0'I      ABEND
            JPLS        SQ4B        [JUMP IF NOT

SQ4A:       JPSR        SRNK        [SHRINK TABLE
```

```
          MDAR'I      CREND      [STILL FULL
          MDX0'I      ABEND
          JPLS        •+2        [JUMP IF NO

          JUMP        SS

SQ4B:     ARX0'F
          ARMD        TXL7C      [RESET  FULL
          MDAR        TRCRD      [FETCH TRACK COORD•
          MDAR'A      CM1H1      [RESET FLAG BITS
          MDAR'0'H    C1         [SET EOL
          MDAR'0      T3         [SET MOVE/DRAW
          ARMD        T2         [SAVE
          MDAR'I      CREND      [FETCH END OF LIST ADDR•
          ARMD        T1
          MDAR'I'N    T1         [FETCH WORD
          MDAR'0'H    C1         [REMOVE EOL
          ARMD'I'N    T1         [RESTORE
          MDAR        T2         [SAVE NEW VECTOR
          ARMD'I'X    T1
          MDAR        T1         [SET NEW END ADDR
          ARMD'I      CREND
          MDX0'I      ABEND      [FULL NOW
          JPLS        SS         [JUMP IF NOT

          MDAR        CFULL
          ARMD        TXL7C      [SET IN MESSAGE
          JUMP        SS

SQ5:      MDAR        C1         [SET DRAW BIT
          JUMP        SQ4D
```

```
SQ9:          MDAR'I       CBLKP        [FETCH BLOCK ADDR
              ARMD         T1
              MDAR'I       T1           [COMPLEMENT DASH
              MDX0'H       C1
              ARMD'I       T1
              JUMP         SR

LTRAN:        MDAR         CBLKP        [LEFT TRANSLATE ROUTINE
              ARMD         TEMP         [GET FWA AND SAVE
              MDAR'I       TEMP
              ARMD         TEMP
              MDAR         AMASK        [MASK AND BRING UP
              MDAR'I'A'X   TEMP         [DX FOR MOD
              ARMD         TEMPT
              MDAR'I       TEMP
              MDAE         XLINC        [MODIFY DX
              MDAR'A'H     AMASK
              MDAR'0       TEMPT        [COMBINE WITH DY
              ARMD'I       TEMP         [PUT BACK IN BLOCK
              ARMD         TEMP1        [ALSO FOR STND ALONE
              JUMP         SF

RTRAN:        MDAR         CBLKP        [RT TRANSLATE ROUTINE
              ARMD         TEMP         [GET FWA AND SAVE
              MDAR'I       TEMP
              ARMD         TEMP
              MDAR         AMASK        [MASK AND BRING UP
              MDAR'I'A'X   TEMP         [DX FOR MOD
              ARMD         TEMPT
              MDAR'I       TEMP
              MDAE         XRINC        [MODIFY DX
              MDAR'A'H     AMASK
```

```
          MDAR'0      TEMPT       [C0MBINE WITH DY
          ARMD'I      TEMP        [PUT BACK IN BL0CK
          ARMD        TEMP1       [ALS0 F0R STND AL0NE
          JUMP        SF

UTRAN:    MDAR        CBLKP       [UP TRANS R0UTINE
          ARMD        TEMP        [GET FWA AND SAVE
          MDAR'I      TEMP
          ARMD        TEMP
          MDAR'I'X    TEMP
          MDAE        UPINC       [M0DIFY DY
          ARMD'I      TEMP        [PUT BACK IN BL0CK
          ARMD        TEMP1       [ALS0 F0R STND AL0NE
          JUMP        SF

DTRAN:    MDAR        CBLKP       [D0WN TRANS R0UTINE
          ARMD        TEMP        [GET FWA AND SAVE
          MDAR'I      TEMP
          ARMD        TEMP
          MDAR'I'X    TEMP
          MDAE        DNINC       [M0DIFY DY
          ARMD'I      TEMP        [PUT BACK IN BL0CK
          ARMD        TEMP1       [ALS0 F0R STND AL0NE
          JUMP        SF

BIG:      MDAR        CBLKP       [MAGNIFY
          ARMD        TEMPS       [GET FWA AND SAVE
          MDAR'I      TEMPS
          ARMD        TEMPS
          MDAR'I      TEMPS
          MDAE        PSCALE      [M0DIFY SCALE
          ARMD'I      TEMPS       [PUT BACK IN BL0CK
```

```
        ARMD        TEMP2       [ALSO FOR STND ALONE
        JUMP        SF

SMALL:  MDAR        CBLKP       [DIMINISH SIZE
        ARMD        TEMPS       [GET FWA AND SAVE
        MDAR'I      TEMPS
        ARMD        TEMPS
        MDAR'I      TEMPS
        MDAE        MSCALE      [MODIFY SCALE
        ARMD'I      TEMPS       [PUT BACK IN BLOCK
        ARMD        TEMP2       [ALSO FOR STND ALONE
        JUMP        SF

ROTAT:  MDAR        ANG         [ROTATION ROUTINE
        JPSR        $SNCOS      [PASS ARG TO SNCOS
        MDAR        CBLKP       [GET FWA OF BLOCK
        ARMD        TEMPR
        MDAR'I      TEMPR
        ARMD        TEMPR
        MDAR'I'X    TEMPR       [INDEX PAST FIRST,
        MDAR'I'X    TEMPR       [SECOND WORDS
ROTA:   ARAR'H                  [NOW BRING UP EACH
        MDAR'A      C1          [DATA WORD AND APPLY
        JPLS        SF          [ROTATION OF COORD
        MDAR'I      TEMPR       [EQUATIONS AS PER
        MDAR'A      EMASK       [CRC HANDBOOK
        ARMD        BITS
        MDAR'I      TEMPR
        MDAR'A'L;   77777JH     [SAVE M/D, AND EOL
        ARMD        XTEMP       [BITS
        MDAR'I      TEMPR
        MDAR'A'L;   77777
```

```
          ARMD           YTEMP
          MDAR           $CØSN
          MPYU           XTEMP
          NØØP
          ARMD           XCØS
          MDAR           $SINE
          MPYL           YTEMP
          NØØP
          ARMD           YSIN
          MDAR           $CØSN
          MPYL           YTEMP
          NØØP
          ARMD           YCØS
          MDAR           $SINE
          MPYU           XTEMP
          NØØP
          ARMD           XSIN
          MDAE           YCØS
          ARAR'H'F
          MDAR'A'L,      77777
          ARMD           XSYC
          MDAR'N         YSIN
          MDAE           XCØS
          MDAR'A'L,      77777JH
          MDAS           XSYC
          MDAR'A         RMASK
          MDAR'Ø         BITS
          ARMD'I         TEMPR
          MDAR'I'X       TEMPR
          JUMP           RØTA
SR:       IFSWITCH (((11,SR1),(15,SR2)))
          JUMP           SS
```

```
SR1:        MDAR'N       TEPFG        [TRACK ENDPØINT MØDE
            JPAN         ST           [JUMP IF NØT

            MDAR         TRCRD        [FETCH CURRENT TRACK CØØRDS.
            MDAR'A       CM1H1        [REMØVE FLAG BITS
            MDAR'Ø       TEPMB        [INSERT CØRRECT ØNES
            ARMD'I       TEPTR        [SET INTØ VECTØR
            JUMP         ST

SR2:        ARXØ'F
            MDAS'H       TRCRD        [CØMPUTE DELTA Y
            MDAS'H'N     SKCRD
            ANXØ         CMO
            MDAS'N       LIMIT
            JPAN         .+2          [SKIP IF NØT EXCEEDED

            JUMP         SR2A         [ELSE, GØ INSERT VECTØR

            ARXØ'F
            MDAS         TRCRD        [CØMPUTE DELTA X
            MDAS'N       SKCRD
            ANXØ         CMO
            MDAS'N       LIMIT
            JPAN         ST           [JUMP IF NØT EXCEEDED

SR2A:       MDAR         C1           [SETUP FØR INSERTING DRAW
            ARMD         T3
            MDAR         TRCRD        [SET NEW SKETCH CØØRD.
            ARMD         SKCRD
            JUMP         SQ4C         [GØ INSERT VECTØR
```

```
SS:       IFSWITCH  (((11,ST)))
          ARXO'F                    [RESET TRACK ENDPOINT MODE
          ARMD          TEPFG
          JUMP          ST


SV:       JPSR          SRNK        [SHRINK TABLE
          MDAR          CBLKP       [FETCH ENTRY ADDR.
          MDAS'F        1           [GET END ADDR
          ARMD          T2
          MDAR'N        NFLAG       [NEW BLOCK
          JPAN          SV1         [JUMP IF NOT

          MDAR          T2          [SETUP ENTRY POINTER
          MDAS'F        1
          ARMD          T3
          MDAR'I        T2          [SET ALLOCATION END
          ARMD'I        T3
          MDAS'F        1
          MDAR'A        AMASK
          ARMD          FREE        [SET FREE STORAGE

SV1:      MDAR'I        T2
          MDAE'N'I      CBLKP       [COMPUTE SIZE
          MDAS'F        2
          MDAR'O'L                  [SETUP COMMAND WORD
          00220/H
          ARMD          SV9A        [SAVE FOR TRANSMIT
          MDAR          CBLKN       [GET BLOCK NO.
          ARMD          SV9A+1      [SAVE
          JPSR          WSWI        [WRITE SWI BUFFER
                        SV9A
```

```
            MDAR        C1          [SET DISPLAY MODE
            ARMD        MODE
            ARMD        CBLKP       [RESET CURRENT POINTER
            ARMD        CURFG       [RESET CURSAR ON

ST:         JPSR        PTRAC       [TRACK PEN
            JUMP        SF          [GO DISPLAY

SV9A:                   0
                        0
                        0


[ PEN TRACKING SUBROUTINE

TI = 300;
MTI = -TI 77777
SI = 40
MSI = -SI 77777

PTRAC:      JUMP        .
            JPSR        PTCK
            MDIR        PTRAC
PTCK:       JUMP        .           [ENTRY
            MDAR'N      CURFG       [CURSAR ON
            ANIR        PTRAC       [RETURN IF NOT

            MDAR'F      NUL         [NULL LIST PIVOT
            ARMD        EOLPV
            MD10'O'L                [TURN ON AVG
            60776√H
```

```
        ARXO'F
        ARMD          LPN2
        ARXO'F                      [CLEAR AR OVF PIVOT
        ARMD          AROPV
        MDAR'F        LPN2          [SETUP PEN IN ERRUPT
        ARMD      .   LPNPV
        MD06'F        0
        MDAR'F        PTPT-1        [SETUP OFFSET TABLE PTR.
        ARMD          PTRCP
        MDAR          TRCRD         [FETCH TRACT COORD
        ARMD          TRCRP         [SAVE FOR UPDATE
        MDAR'A        CM1           [MASK OUT DRAW
        MDAR'O'H      C1            [SET EOL
        JPSR          LOD5
        0; 0; 0; 0; 0;
        JPSR          LOD5
        MDAR          TRCRD         [SET FOR DRAWING DOT
        MDAR'O        C1H1
        NOOP
        NOOP
        JPSR          LOD5
        MDAR          TRCRD         [FETCH COORD
        MDAE'L                      [MOVE TO UPPER RIGHT
        400/H         400
        MDAR'A        CM1           [REMOVE DRAW
        MDAR'O'H      C1            [SET EOL
        NOOP
        NOOP
        JPSR          LOD5
IREPEAT Q,(7740000401,7740077401,40077401,40000401)
        MDAR          TRCRD         [FETCH COORDS.
        MDAE'L                      [OFFSET
```

```
                Q
                MDAR'O          C1H1           [SET DRAW AND EOL
                NOOP
                NOOP
                NOOP
                ARAR'X          PTRCP
                JPSR            LOD5
        ENDI
                NOOP
                NOOP
                NOOP
                NOOP
                MDAR            LPN2
                JPLS            PTCK9
                MDAR'F          LPN22
                ARMD            LPNPV
                MDAR            TRCRD
                MDAE'L
                1750JH          500
                MDAR'A          CM1
                MDAR'O'H        C1
                JPSR            LOD5
        NOCARRET
        IREPEAT Q,(115201523,2033,7662401523,7603000501,
                7603077301,7662476255,75745,115276255,
                175077301,175000501);
        CARRET
                NOOP
                NOOP
                NOOP
                NOOP
                MDAR            TRCRD
```

```
               MDAE'L
               Q
               MDAR'O          C1H1
               ARAR'X          PTRCP
               JPSR            LOD5
ENDI
               NOOP
               NOOP
               NOOP
               NOOP
PTCK9:         MDAR            TRCRP
               ARMD            TRCRD
               MDAR'F          NULX
               MD10'A'L                        [RESET AVG
               1000JH
               ARMD            LPNPV           [NULL LIGHT PEN PIVOT
               MDIR            PTCK            [RETURN
                               0
PTPT:          SI;MSIJH;MSI;SIJH

               145101010;57601777;7671201777;7632701012
               7603077700;7632776776;7671276000;57676000
               145076770;175077700
               NOOP; NOOP

[ INTERRUPT SUBR. FOR LIGHT PEN

LPN2:                          0
               ARMD            LPN2A           [SAVE (AR)
               MDAR            TRCRP           [ADJUST POSITION
               MDAE'I          PTRCP
               ARMD            TRCRP           [SAVE
```

```
                MDAR            LPN2A           [RESTORE (AR)
                JUMP'I          LPN2            [RETURN AND UNLOCK

LPN22:                          0
                ARMD            LPN2A
                MDAR            TRCRP
                MDAE'I          PTRCP
                ARMD            TRCRP
                MDAR'F          NULX
                ARMD            LPNPV
                MDAR            LPN2A
                JUMP'I          LPN22
LPN2A:                          0
PTRCP:                          0
TRCRP:                          0

LOD5:           JUMP            .
                ARMD'L)         0
                MD05            .-1
                MDIR            LOD5
[ COMMUNICATIONS SUBROUTINES..........

[WRITE SWI BUFFER SUBR.

WSWI:           JUMP            .               [ENTRY
                MDAR            DBUG
                JPLS            WSWIX
                MDAR'I          WSWI            [FETCH AGT ADDRESS
                ARMD            .+4             [SAVE
                JPSR            $ROWFW          [WRITE
                                -0
                                $SWIA
```

```
                                  0
                                  3
                 OPIO             43005             [GIVE INTERRUPT
WSWIX:
                 MDIR'X           WSWI              [RETURN


[ FILL SWI BUFFER SUBR.

FSWI:            JUMP             .                 [ENTRY
                 MDAR             DBUG
                 JPLS             FSWIX
                 ARXO'F                             [RESET COMMAND FLAG
                 ARMD             $GTWD1
                 JPSR             $FINSH            [SET ACCESS BIT
                 MDAR'I           FSWI              [FETCH ADG.
                 ARMD             .+2               [SAVE FOR WSWI CALL
                 JPSR             WSWI              [WRITE SWI BUFFER
                                  0
FSWIX:
                 MDIR'X           FSWI              [RETURN


[SWI BLOCKS TO BE TRANSMITTED:

1BLK:            100/H
2BLK:            110/H
3BLK:            120/H
4BLK:            130/H
5BLK:            140/H
6BLK:            200/H
7BLK:            10/H
```

```
[ TEXT MODE MESSAGES..........

[DISPLAY MODE

TXL1:           1√H


[TEXT WAIT MODE

TXL2:
                ORIGIN
                P (T.,E.,X.,T.)
                P (40,E.,D.,I.)
                P (T.,40,R.,E.)
                P (Q.,U.,E.,S.)
                P (T.,,,,1)


[GRAPHICS WAIT MODE

TXL3:
                ORIGIN
                P (G.,R.,A.,P.)
                P (H.,I.,C.,S.)
                P (40,E.,D.,I.)
                P (T.,40,R.,E.)
                P (Q.,U.,E.,S.)
                P (T.,,,,1)
```

```
[TEXT SELECT MØDE

TXL4:
                ØRIGIN
                P (T.,E.,X.,T.)
                P (40,B.,L.,Ø.)
                P (C.,K.,40,S.)
                P (E.,L.,E.,C.)
                P (T.,40,M.,Ø.)
                P (D.,E.,40,40)
                P (40,40,40,40)
                P (40,40,40,40)
                P (40,40,B.,L.)
                P (Ø.,C.,K.,40)
TXL4A:          O
                1√H


(GRAPHICS SELECT MØDE

TXL5:
                ØRIGIN
                P (G.,R.,A.,P.)
                P (H.,I.,C.,S.)
                P (40,B.,L.,Ø.)
                P (C.,K.,40,S.)
                P (E.,L.,E.,C.)
                P (T.,40,M.,Ø.)
                P (D.,E.,40,40)
                P (40,40,40,40)
                P (40,40,B.,L.)
                P (Ø.,C.,K.,40)
```

```
TXL5A:          0
                1√H


[TEXT EDIT MODE

TXL6:
                ORIGIN
                P (T.,E.,X.,T.)
                P (40,E.,D.,I.)
                P (T.,40,M.,O.)
                P (D.,E.,40,40)
                P (40,40,B.,L.)
                P (O.,C.,K.,40)
TXL6A:          0
                P (40,40,40,S.)
                P (I.,Z.,E.,40)
TXL6B:          0
                P (40,40,40,X.)
                P (40)
TXL6C:          0
                P (40,40,40,Y.)
                P (40)
TXL6D:          0
                P (40,40,40,40)
TXL6E:          0
                1√H


[GRAPHICS EDIT MODE

TXL7:
```

```
                ORIGIN
                P (G.,R.,A.,P.)
                P (H.,I.,C.,S.)
                P (40,E.,D.,I.)
                P (T.,40,M.,O.)
                P (D.,E.,40,40)
                P (40,40,B.,L.)
                P (O.,C.,K.,40)
TXL7A:          0
                P (40,40,40,40)
                P (C.,U.,R.,S.)
                P (A.,R.,40)
TXL7B:          P (O.,F.,F.)
                P (40,40,40,40)
TXL7C:          0
                1JH


ERMSG:          P (13,106.,11,140)
                P (22,37,S.,T.)
                P (O.,R.,A.,G.)
                P (E.,40,F.,U.)
                P (L.,L.,,,1)
[GET ASCII CHARACTERS FOR VALUE SUBR.

GASCI:          JUMP          .
                MDAR'A        CMASK         [MASK VALUE
                ARLS          1
                DIVI          10.           [DIVIDE BY 10
                NOOP
                ARMD'H        GAS1          [SAVE REMAINDER
                JPLS          .+2           [SKIP IF NOT ZERO
```

```
          MDAR'N          C20           [SET -20 FØR SPACE

          MDAS'F          60            [CREATE ASCII CHARACTER
          MDAR'A          CMASK
          ARMD'K          GAS2          [SAVE
          ARXØ'F
          MDAS            GAS1          [FETCH REMAINDER
          ARRS            1
          MDAS'F          60            [CREATE CHARACTER
          MDAR'A          CMASK
          MDAR'Ø'B        GAS2          [CØMBINE FIRST CHARACTER
          ARAR'B'F
          MDIR            GASCI         [RETURN

GAS1:                     0
GAS2:                     0


[GET SIZE NUMBER SUBRØUTINE

GSIZ:     JUMP            .
          JPLS            .+3           [SKIP IF NØT MIDDLE SIZE

          MDAR            C2            [SET SIZE = 2
          MDIR            GSIZ

          MDXØ'F          (22√B)        [SMALL
          JPLS            .+3           [SKIP IF NØT

          MDAR            C1            [SET SIZE = 1
          MDIR            GSIZ
```

```
            MDAR        C3          [SET SIZE = 3
            MDIR        GSIZ

[GET INTENSITY NUMBER SUBR.

GINT:       JUMP        .
            JPLS        .+3         [SKIP IF NØT MIDDLE

            MDAR        C2          [SET INT. = 2
            MDIR        GINT

            MDXØ'F      (36√B)      [DIM
            JPLS        .+3         [SKIP IF NØT

            MDAR        C1          [SET INT. = 1
            MDIR        GINT

            MDAR        C3          [SET INT. = 3
            MDIR        GINT

[ RESET SELECTED VECTØR SUBR.

RSV:        JUMP        .
            MDAR'N      SVFLG       [VECTØR SELECTED
            ANIR        RSV         [RETURN IF NØT

            ARMD        SVFLG       [RESET FLAG
            MDAR'I      SVPTR       [FETCH VECTØR
            MDAR'Ø      C1          [SET DRAW BIT ØN
```

```
                ARMD'I          SVPTR           [RESTØRE
                MDAR'F          DUMMY
                ARMD            SVPTR           [SETUP DUMMY PØINTER
                MDIR            RSV             [RETURN

[ PEN INTERRUPT SUBRØUTINE

PENS:           JUMP            .
                ARMD            ARSAV           [SAVE (AR)
                MDAR            CURFG           [CURSAR ØN
                JPAN            PENS2           [RETURN IF YES

                JPSR            RSV             [RESET ANY SELECTED VECTØR
                MDAR            EØVPV           [FETCH VECTØR PIVØT
                MDAS'F'N        2               [BACK UP
                ARMD            SVPTR           [SAVE

PENS1:          MDAR'I'X'N      SVPTR           [FETCH VECTØR
                MDAR'A'F        1               [DRAW
                JPLS            .-2             [REPEAT IF NØT

                MDAR'I          SVPTR           [FETCH VECTØR
                MDXØ            C1              [MAKE IT A MØVE
                ARMD'I          SVPTR           [RESTØRE
                ARMD'Ø          SVFLG           [SET SELECTED VECTØR FLAG
PENS2:          MDAR            ARSAV           [RESTØRE (AR)
                JUMP'I          PENS            [RETURN AND UNLØCK

ARSAV:                          0
```

```
[PUSHDOWN CHARACTER LIST SUBROUTINE

PSHDN:      JUMP        .
            MDAR        CPOS            [SETUP POINTERS
            ARMD        T1
            MDAS'F'N    1
            ARMD        T2
            MDX0'F      CRLIN+98.       [END
            JPLS        .+2             [SKIP IF NOT

            JUMP        PSH2

PSH1:       MDAR'I'X    T1              [MOVE CHARACTER DOWN
            ARMD'I'X    T2
            MDAR        T1              [END OF MOVE
            MDX0'F      CRLIN+99.
            JPLS        PSH1            [REPEAT IF NOT

PSH2:       ARX0'F
            ARMD        CRLIN+99.       [SET LAST CHAR = NULL
            ARMD        TXL6E           [REMOVE  FULL
            MDIR        PSHDN           [RETURN


[SHRINK VECTOR LIST SUBROUTINE

SRNK:       JUMP        .
            JPSR        RSV             [RESET ANY SELECTED VECTORS
            MDAR'I      CBLKP           [GET START OF BLOCK
            ARMD        SRN9            [SET POINTERS
            ARMD        SRNO
            ARX0'F                      [RESET MOVE FLAG
```

```
        ARMD          SRN8
        MDAR'X        SRN0          [INDEX TØ GET 2ND
        MDAR'X        SRN9          [WØRDN

SRN1:   MDAR'I'X      SRN0
        MDAR'A'F      1             [DRAW
        JPLS          SRN2          [JUMP IF YES

        MDAR          SRN8          [SARED MØVE
        JPLS          .+2           [SKIP IF YES

        JUMP          .+3           [IGNØRE IF NØT


        ARXØ'F
        ARMD          TXL7C         [REMØVE  FULL
        MDAR          SRN0          [SAVE MØVE PØINTER
        ARMD          SRN8

SRN3:   MDAR          SRN0          [END ØF LIST
        MDXØ'I        CREND
        JPLS          SRN1          [JUMP IF NØT

        JPSR          SRNS          [CØPY ANY MØVE
        MDAR          SRN9          [SET NEW CURRENT END
        ARMD'I        CREND
        MDIR          SRNK          [RETURN

SRN2:   JPSR          SRNS          [CØPY ANY MØVE
        MDAR'I        SRN0          [CØPY THIS DRAW
        ARMD'I'X      SRN9
        JUMP          SRN3          [GØ CHECK END
```

```
SRN0:                       0
SRN9:                       0
SRN8:                       0

SRNS:           JUMP        .
                MDAR        SRN8        [ANY MOVE SAVED
                JPLS        .+2         [SKIP IF YES

                MDIR        SRNS        [ELSE, RETURN

                MDAR'I      SRN8        [FETCH MOVE
                ARMD'I'X    SRN9        [SAVE
                ARX0'F                  [RESET MOVE FLAG
                ARMD        SRN8
                MDIR        SRNS        [RETURN


[ DISPLAY TEXT LINE SUBROUTINE

DTEXT:          JUMP        .
                MDAS'F'N    1
                ARMD        LCGAD       [SAVE LCG PIVOT
                MDIC'A      CM14        [RESET LCG
                MDAR'F      LCGS        [SETUP END LIST PIVOTS
                ARMD        LCGEA
                ARMD        LCGEB
                MD07        ZERO        [RESET DX, DY
                MD10'O'L                [TURN ON 2D AND SCOPES
                40760/H
                MD05        ZERO        [MOVE BEAM TO CENTER
                ARX0'F
```

```
          ARMD           T1             [RESET END FLAG
          MDIC'Ø         C10            [TURN ØN LCG
          MD10'A'L                      [RESET AVG
          -40000√H
          MDAR'X         T1             [FETCH END FLAG
          ANIR           DTEXT          [RETURN IF DØNE
          JPLS           .-2            [REPEAT IF NØT TIME-ØUT

          MDIC'A         CM14           [RESET LCG
          MDIR           DTEXT          [RETURN


[ LCG INTERRUPT SUBRØUTINE

LCGS:     JUMP           .
          MDIC'A         CM14           [RESET LCG
          ARMD'Ø         T1             [SET END FLAG
          JUMP'I         LCGS           [RETURN AND UNLØCK


[ FRAME CLØCK INTERRUPT SUBRØUTINE

CLKS:     JUMP           .
          ARMD           CLKAR          [SAVE (AR)
          MDAR'H'X       FTIME          [INDEX TIMER
          JPAN           .+2            [SKIP IF NØT THREE TICKS

          ARMD           FFLAG          [SET FRAME FLAG IF DØNE

          MDAR           CLKAR          [RESTØRE (AR)
          JUMP'I         CLKS           [RETURN AND UNLØCK
```

```
CLKAR:                  0

[ AVG END OF LIST INTERRUPT SUBROUTINE

EOLS:           JUMP            .
                ARMD'O          DSDON           [SET DONE FLAG
                JUMP'I          EOLS            [RETURN AND UNLOCK
[ CHECK SWITCH CHANGE SUBROUTINE


IFC:            JUMP            .               [ENTRY
                MDAR            CNGS            [FETCH CHANGED SWITCHES

IFC2:           ARMD            IFC1            [SAVE
                MDAR'I          IFC             [FETCH ARG.
                ARAR'X          IFC
                JPLS            .+2             [SKIP IF ARGUMENT

                MDIR            IFC             [ELSE, RETURN

                ARMD            IFC9            [SAVE
                ARAR'F'K                        [GET NO.
                MDAR'A          VMASK
                MDAE'L                          [CREATE SHIFT
                ARLS            13.
                ARMD            .+3             [SAVE
                MDAR'L'N                        [FETCH SWITCHES
IFC1:                           0
                NOOP
                NOOP
                NOOP
```

```
                JPAN            IFC2+1          [REPEAT IF NOT ON

                JUMP'I          IFC9            [ELSE, DISPATCH

IFC9:                           0



[ IF SWITCH ON SUBROUTINE

IFS:            JUMP            .               [ENTRY
                MDAR            IFS             [SAVE ENTRY
                ARMD            IFC
                MDAR            NEWS            [FETCH SWITCHES ON
                JUMP            IFC2            [GO PROCESS



[ NULL SUBROUTINES

NUL:            JUMP            .
                JUMP'I          NUL

NULX:           JUMP            .
                JUMP'I          NULX
[ TABLES

TBUFF:          LOC (.+51.)                     [COMM. BUFFER

                P (0,0,0,10)
```

```
CRLIN:          LOC (.+100.)                    [CURRENT TEXT LINE BUFFER
                P (0,0,0,0,1)

STBL:                           22              [SIZE TABLE
                                0
                                23

ITBL:                           36              [INTENSITY TABLE
                                0
                                37

TBLK:                           .               [TEXT BLOCK LIST
                LOC             (.+244.)
TBLKE:                          .

GBLK:                           .               [GRAPHICS BLOCK LIST
                LOC             (.+154.)
GBLKE:                          .

TXLIN:                          TXL1            [MODE MESSAGE TABLE
                                TXL2
                                TXL3
                                TXL4
                                TXL5
                                TXL6
                                TXL7


[ CONSTANTS
ANG:                            1000
RMASK:                          7777677776
EMASK:                          0000100001
```

```
XRINC:                         00070√H
XLINC:                         77707√H
DNINC:                         77707
UPINC:                         00070
PSCALE:                        00070√H
MSCALE:                        77706√H
SCMASK:                        77760
AMASK:                         77777
CMASK:                         177
VMASK:                         77
ZER0:                          0
C1:                            1
C2:                            2
C3:                            3
C4:                            4
C5:                            5
C6:                            6
C7:                            7
C10:                           10
C20:                           20
C40:                           40
C137:                          137
C26.:                          26.
C50.:                          50.
CB15:                          40000
C1H1:           1√H            1
CM0:                           -0
CM1:                           -1
CM1H1:          77776√H        77776
CM14:                          -14
CM40:                          -40
CFULL:          P (F.,U.,L.,L.)
C0FF:           P (0.,F.,F.)
```

```
CON:            P (O.,N.)
M1629:                          37777
M0010:          77760VH
M1528:                          77776
M1828:                          7776
M2228:                          376
LIMIT:                          100
[ VARIABLES
TEMPT:                          0
TEMP3:                          0
CNT:                            0
TEMP:                           0
TEMP1:                          0
TEMPS:                          0
TEMP2:                          3777610000
BITS:                           0
XTEMP:                          0
YTEMP:                          0
XCOS:                           0
YSIN:                           0
XSIN:                           0
YCOS:                           0
XSYC:                           0
TEMPR:                          0

T1:0;T2:0;T3:0;T4:0;T5:0;T6:0;T7:0;T10:0
TPTR:                           0
CPTR:                           0
SVPTR:                          0
FREE:                           0
TBLKN:                          0
GBLKN:                          0
```

```
CBLKN:                          0
CBLKP:                          0
MØDE:                           0
TØLDS:                          0
ØLDS:                           0
NEWS:                           0
CNGS:                           0
FTIME:                          0
FFLAG:                          0
ØKFLG:                          0
NFLAG:                          0
DSDØN:                          0
ØLKP:                           0
CPØS:                           0
CURCT:                          0
REPCT:                          0
CFLAG:                          0
CHAR:                           0
CURFG:                          0
SVFLG:                          0
TEPFG:                          0
CREND:                          0
ABEND:                          0
TRCRD:                          0
TEPTR:                          0
TEPMB:                          0
SKCRD:                          0
DUMMY:                          0
DBUG:                           0
IFPASS1  TYPEØUT '
LENGTH = '  TYPEØCT (.);   ENDC
TERMINATE
```

LIST OF REFERENCES

1. Manchover, C., "The Intelligent Terminal", in *Pertinent Concepts in Computer Graphics*, Faiman M., and Nievergelt, J., eds., University of Illinois Press, Urbana, Ill., 1969.

2. Streit, E. "VIP: A Conversational System For Computer-Graphics", in *Pertinent Concepts in Computer Graphics*, Faiman, M., and Nievergelt, J., eds., University of Illinois Press, Urbana, Ill., 1969.

3. Notely, M.G., "A Graphical Picture Drawing Language", *The Computer Bulletin*, v. 14, pp. 68-74, March, 1970.

4. Beans, J., "GPGL: A Model Interactive, General Purpose Graphic Language", Thesis, U.S. Naval Postgraduate School, Monterey, Cal. 1971.

5. Kulsrud, H.E., "A General Purpose Graphic Language", Communications of the ACM V. 11, pp 247-254, April, 1968.

6. DeLaura, R.D., *Electrical Engineering Computer Laboratory Manual for the Fortran User*, U.S. Naval Postgraduate School, Monterey, Cal., 1972.

7. Hamilton, J.A., *A Survey of Data Structures for Interactive Graphics*, The Rand Corporation, RM-6145-ARPA, April, 1970.

8. Beckermeyer, R.L., "Interactive Graphics Consoles-Environment and Software", in *AFIPS Conference, Proceedings 1970 Fall Joint Computer Conference*, v. 37, AFIPS Press, 1970.

9. Newman, W., "A System for Interactive Graphical Programming", in *AFIPS Conference Proceedings 1968 Spring Joint Computer Conference*, v. 32, Thompson Book Company, 1968.

10. Williams, R., "A Survey of Data Structures for Computer Graphics Systems;, *Computer Survey*, v. 3, no. 1, March, 1971.

11. Licklider, J.C.R., "Man-Computer Symbiosis", IRE, Transactions HFE, pp. 4-11, March, 1960.

12. Sutherland, I.E., "Sketchpad: A Man-Machine Graphical Communication System", in *AFIPS Conference Proceedings 1963 Spring Joint Computer Conference*, 1963.

13. Sutherland, I.E., "Computer Graphics", Datamation, pp. 22-27, May, 1966.

14. Fetter, W.A., Computer Graphics in Communication, McGraw-Hill Book Company, New York, 1965.

15. Baskin, H.B., and Morse, S.P., "A Multilevel Modeling Structure for Interactive Graphic Design", IBM SYSTEMS JOURNAL, v. 7, no 3, 4, pp. 218-228, 1968.

16. Hornbuckle, G.D., "The Computer Graphics User/Machine Interface", IEEE Transactions on Human Factors in Electronics, HFE-8 (1), March, 1967.

17. SDS SYMBOL AND META-SYMBOL, Scientific Data Systems Reference Manual, Scientific Data Systems, Santa Monica, Cal. 1967.

18. SDS 9300 COMPUTER, Scientific Data Systems Reference Manual, Scientific Data Systems, Santa Monica, Cal. 1967.

19. SDS Real-Time MONITOR, Scientific Data Systems Reference Manual, Scientific Data Systems, Santa Monica, Cal. 1967.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Professor George A. Rahe Department of Electrical Engineering Naval Postgraduate School Monterey, California	2
3.	Lieutenant Richard F. Ashford, Jr. Long Beach Naval Shipyard Long Beach, Cal. 90801	1
4.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

RADIK: An Interactive Graphics and Text Editor

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

Master's Thesis; June 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

Richard F. Ashford, Jr.

6. REPORT DATE

June 1972

7a. TOTAL NO. OF PAGES

137

7b. NO. OF REFS

19

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

RADIK is an interactive graphics and text editing system designed for use with an ADAGE AGT/10 graphics computer, either in a stand-alone mode, or in conjunction with an XDS 9300 computer. The thesis presents an overview of desirable attributes and capabilities of an interactive graphics display system. A description is given of the graphics display system presently in use at the Naval Postgraduate School Computer Laboratory, along with its apparent deficiencies. Objectives for an improved graphics and text editor are presented, in addition to results achieved and problems encountered while designing RADIK. A brief summary of results and applications is presented and implementation of RADIK is proposed. Computer programs developed during the work are appended for reference.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Interactive Graphics						
Computer Graphics						
Graphics Displays						
Computer-aided Design						
Interactive Programs						

134713

Thesis
A795 Ashford
c.1 RADIK: an interactive graphics and text editor.